**MySQL Master/Slave setup and install from Manual**

**note: I create 2 instances MySQL for Master MySQL and Slave MySQL. Following these steps:**

**+ Install MySQL [ ref :** [**Training 3th**](https://docs.google.com/document/d/1ToqLQXgAo_46CI7xAXjpmcT32x38fxC9RCdZC1GdMyQ/edit?tab=t.0) **]**

**# Install MySQL 5.7.28 (install cmake) on a Ubuntu server**

sudo apt update

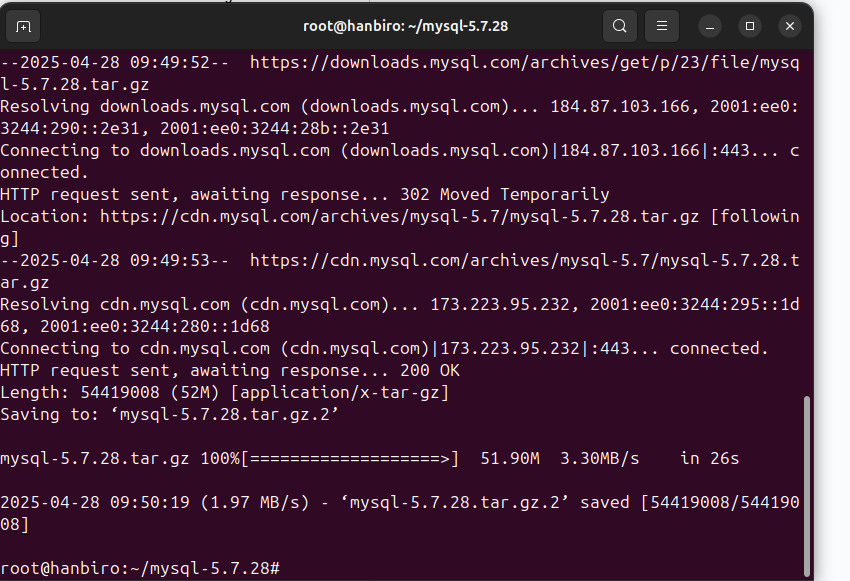
sudo apt install -y build-essential cmake libncurses5-dev libaio-dev bison zlib1g-dev pkg-config libtirpc-dev libssl-dev

cd ~

wget https://downloads.mysql.com/archives/get/p/23/file/mysql-5.7.28.tar.gz

tar -xzf mysql-5.7.28.tar.gz

cd mysql-5.7.28



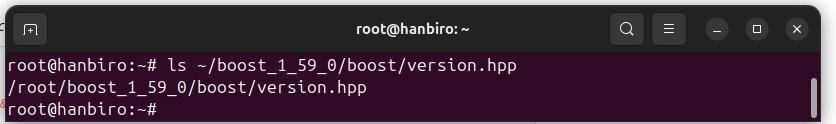
* download and unzip Boost 1.59.0:

cd ~

wget https://sourceforge.net/projects/boost/files/boost/1.59.0/boost\_1\_59\_0.tar.gz/download -O boost\_1\_59\_0.tar.gz

tar -xzf boost\_1\_59\_0.tar.gz

check file: ls ~/boost\_1\_59\_0/boost/version.hpp



* compile OpenSSL 1.1.1 (required since MySQL 5.7 does not support OpenSSL 3):

cd ~

wget https://www.openssl.org/source/openssl-1.1.1w.tar.gz

tar -xzf openssl-1.1.1w.tar.gz

cd openssl-1.1.1w

./config --prefix=/opt/openssl-1.1.1 --openssldir=/opt/openssl-1.1.1 no-shared

make -j$(nproc)

sudo make install

* download MySQL source code and create ‘build’ directory:

cd ~

wget https://downloads.mysql.com/archives/get/p/23/file/mysql-5.7.28.tar.gz

tar -xzf mysql-5.7.28.tar.gz

cd mysql-5.7.28

mkdir build && cd build

* run cmake with full configuration and fix errors:

**cmake .. \**

**-DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql \**

**-DWITH\_SSL=/opt/openssl-1.1.1 \**

**-DOPENSSL\_INCLUDE\_DIR=/opt/openssl-1.1.1/include \**

**-DOPENSSL\_LIBRARIES=/opt/openssl-1.1.1/lib/libssl.a\;/opt/openssl-1.1.1/lib/libcrypto.a \**

**-DDOWNLOAD\_BOOST=0 \**

**-DWITH\_BOOST=~/boost\_1\_59\_0 \**

**-DCMAKE\_C\_FLAGS="-I/usr/include/tirpc" \**

**-DCMAKE\_CXX\_FLAGS="-I/usr/include/tirpc" \**

**-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \**

**-DWITH\_PARTITION\_STORAGE\_ENGINE=1 \**

**-DWITH\_FEDERATED\_STORAGE\_ENGINE=1 \**

**-DWITH\_EXTRA\_CHARSETS=all \**

**-DWITH\_ZLIB=system \**

**-DENABLED\_LOCAL\_INFILE=1**

**\*Note:**

* use OpenSSL manually due to OpenSSL 3 incompatibility
* Boost must be available, do not use -DDOWNLOAD\_BOOST=1 because the old download link does not work
* add flag -I/usr/include/tirpc to handle rpc/rpc.h errors
* -all-static => --static
* build and config:

make -j$(nproc)

sudo make install

* **Master MySQL running on port 3307, data/log at /home/MYSQLDATA/master & /home/MYSQLLOGS/master**
* **Slave MySQL running on port 3308, data/log at /home/MYSQLDATA/slave & /home/MYSQLLOGS/slave**
* **Using MySQL 5.7.28 from source at /usr/local/mysql**
* **Use different sockets to avoid conflicts**
* create Directory, Authorize for Master/Slave:

sudo mkdir -p /home/MYSQLDATA/master /home/MYSQLDATA/slave

sudo mkdir -p /home/MYSQLLOGS/master /home/MYSQLLOGS/slave

sudo mkdir -p /etc/mysql/master /etc/mysql/slave

sudo chown -R mysql:mysql /home/MYSQLDATA /home/MYSQLLOGS

* CREATE CONFIGURATION master FILE **sudo nano /etc/mysql/master/my.cnf**

**[mysqld]**

**server-id=1**

**port=3307**

**datadir=/home/MYSQLDATA/master**

**socket=/tmp/mysql\_master.sock**

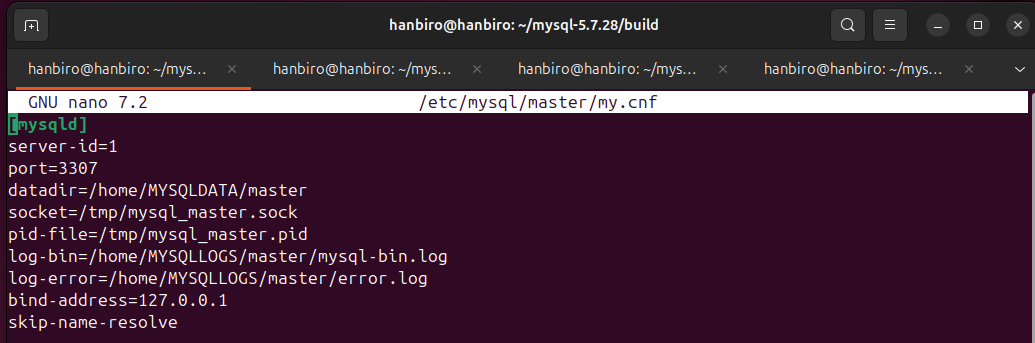
**pid-file=/tmp/mysql\_master.pid**

**log-bin=/home/MYSQLLOGS/master/mysql-bin.log**

**log-error=/home/MYSQLLOGS/master/error.log**

**bind-address=127.0.0.1**

**skip-name-resolve**



* CREATE CONFIGURATION slave FILE **sudo nano /etc/mysql/slave/my.cnf**

**[mysqld]**

**server-id=2**

**port=3308**

**datadir=/home/MYSQLDATA/slave**

**socket=/tmp/mysql\_slave.sock**

**pid-file=/tmp/mysql\_slave.pid**

**log-bin=/home/MYSQLLOGS/slave/mysql-bin.log**

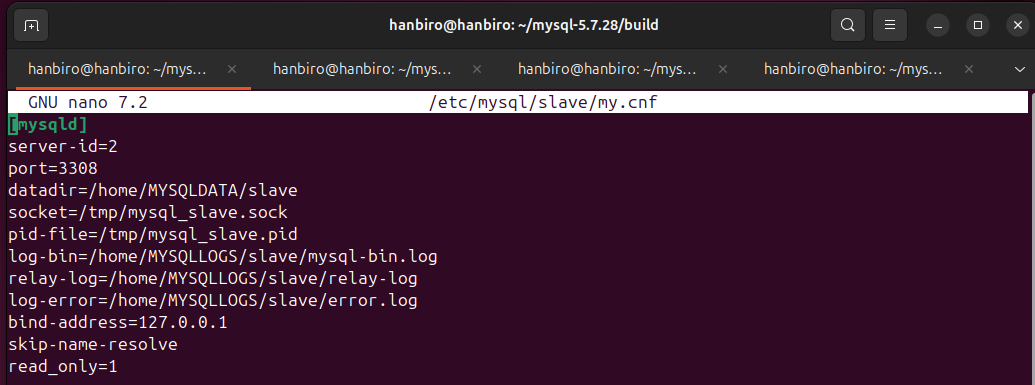
**relay-log=/home/MYSQLLOGS/slave/relay-log**

**log-error=/home/MYSQLLOGS/slave/error.log**

**bind-address=127.0.0.1**

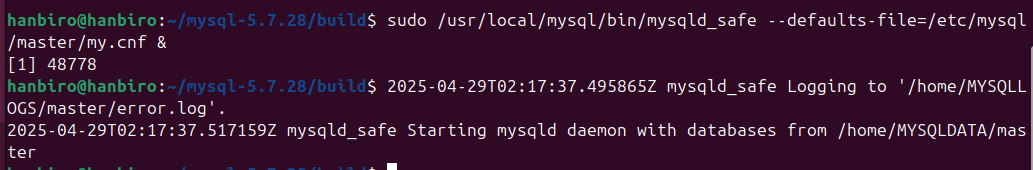
**skip-name-resolve**

**read\_only=1**

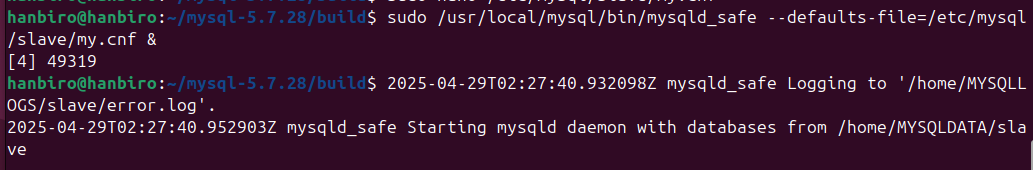


* initialize data for each instance

sudo /usr/local/mysql/bin/mysqld\_safe --defaults-file=/etc/mysql/master/my.cnf &



sudo /usr/local/mysql/bin/mysqld\_safe --defaults-file=/etc/mysql/slave/my.cnf &



* open new Terminal, connect to **Master** instance. create database, user, and show status:

/usr/local/mysql/bin/mysql -uroot -S /tmp/mysql\_master.sock

CREATE DATABASE replicated\_db;

CREATE USER 'replica'@'%' IDENTIFIED BY 'replica\_pass';

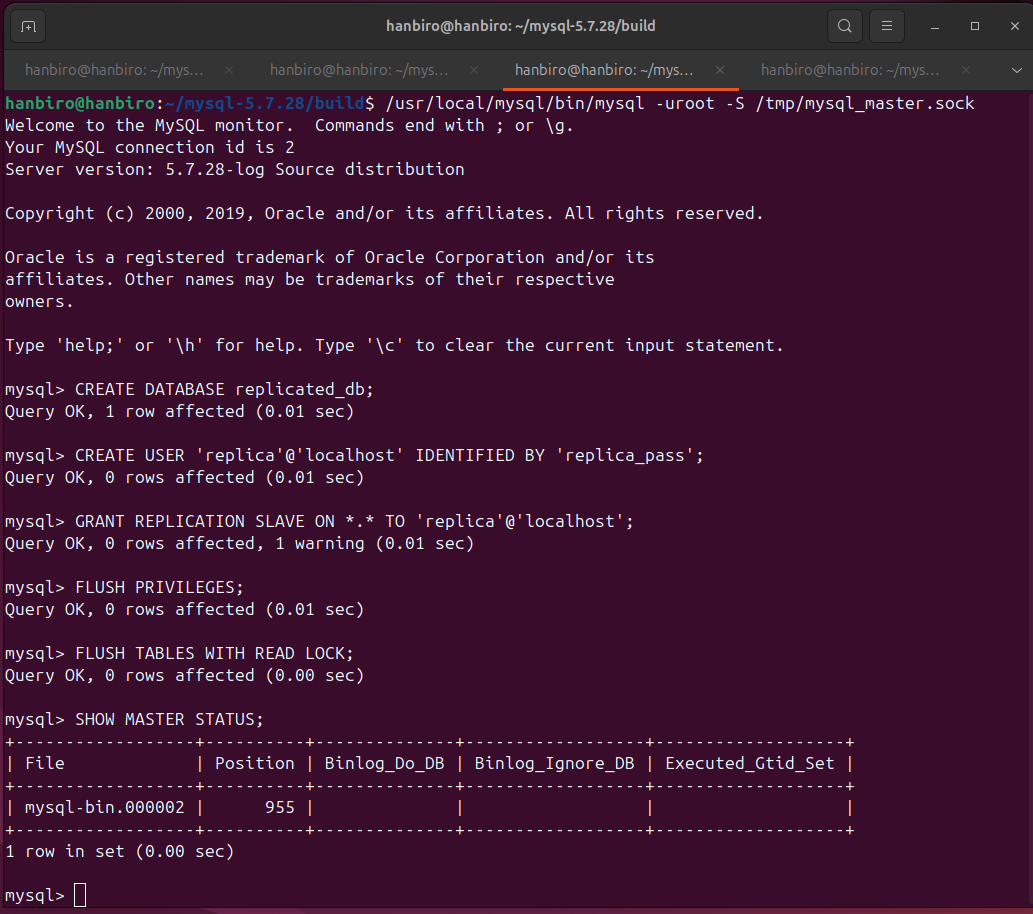
GRANT REPLICATION SLAVE ON \*.\* TO 'replica'@%;

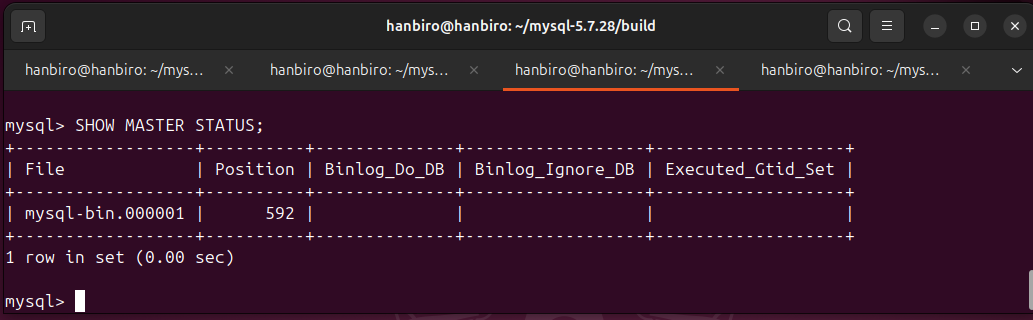
FLUSH PRIVILEGES;

FLUSH TABLES WITH READ LOCK;

SHOW MASTER STATUS;

\*save File (mysql-bin.000001) and Position (592). Do not close this connection until the slave is configured (because the tables are LOCKED)





* open new Terminal, connect to **Slave** instance. Config with Master’s parameter, start and show status of Slave:

/usr/local/mysql/bin/mysql -uslave\_test\_user -p -S /tmp/mysql\_slave.sock

CHANGE MASTER TO

MASTER\_HOST='**127.0.0.1**',

MASTER\_PORT=**3307**,

MASTER\_USER=**'replica'**,

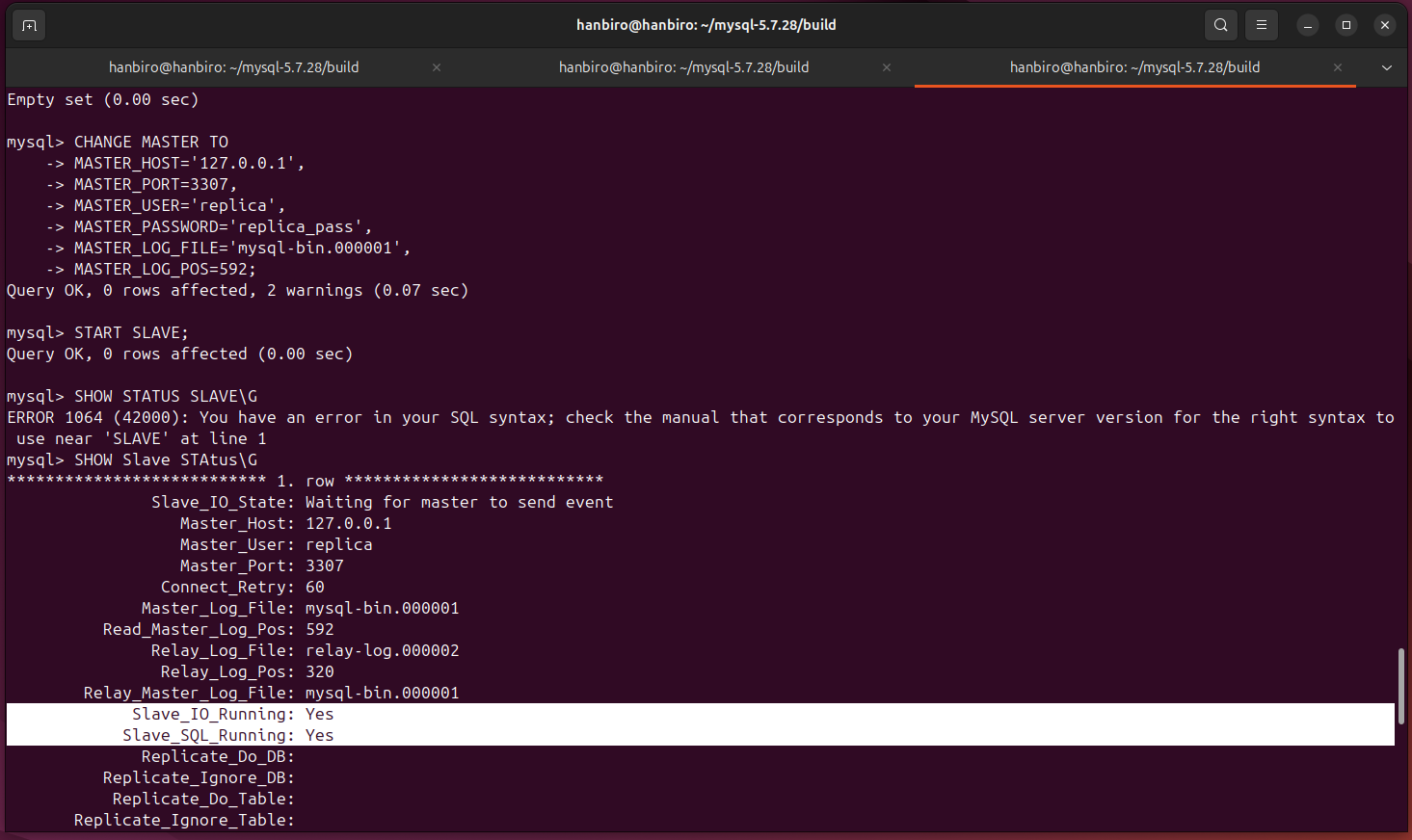
MASTER\_PASSWORD='**replica\_pass**',

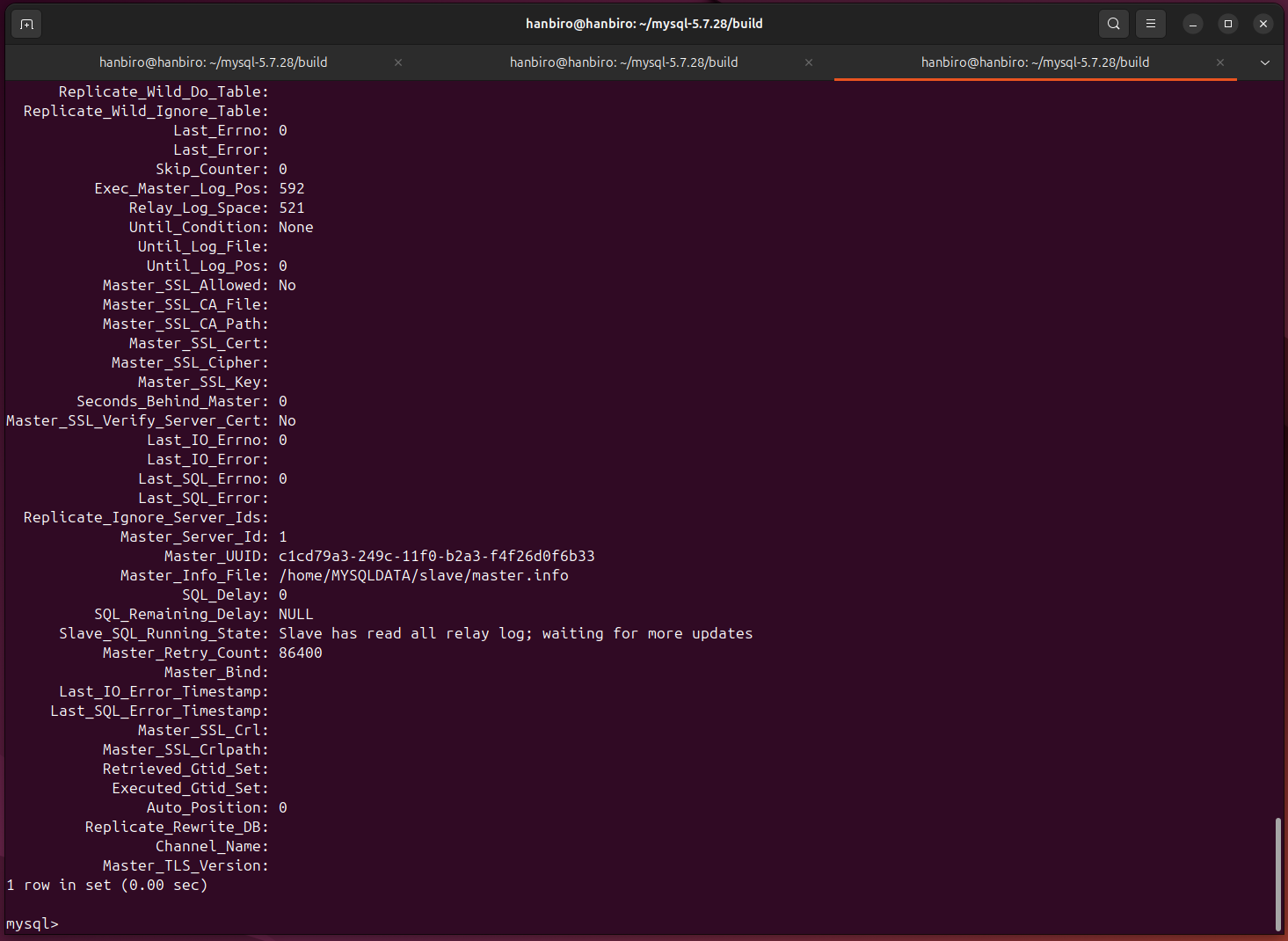
MASTER\_LOG\_FILE='**mysql-bin.000001**',

MASTER\_LOG\_POS=**592**;

START SLAVE;

SHOW SLAVE STATUS\G





**[ TEST CRUD ]**

* **CREATE Table and Data on Master**

connect to Master MySQL:

/usr/local/mysql/bin/mysql -uroot -S /tmp/mysql\_master.sock

create ‘test\_table’ table in ‘replicated\_db’ database:

USE replicated\_db;

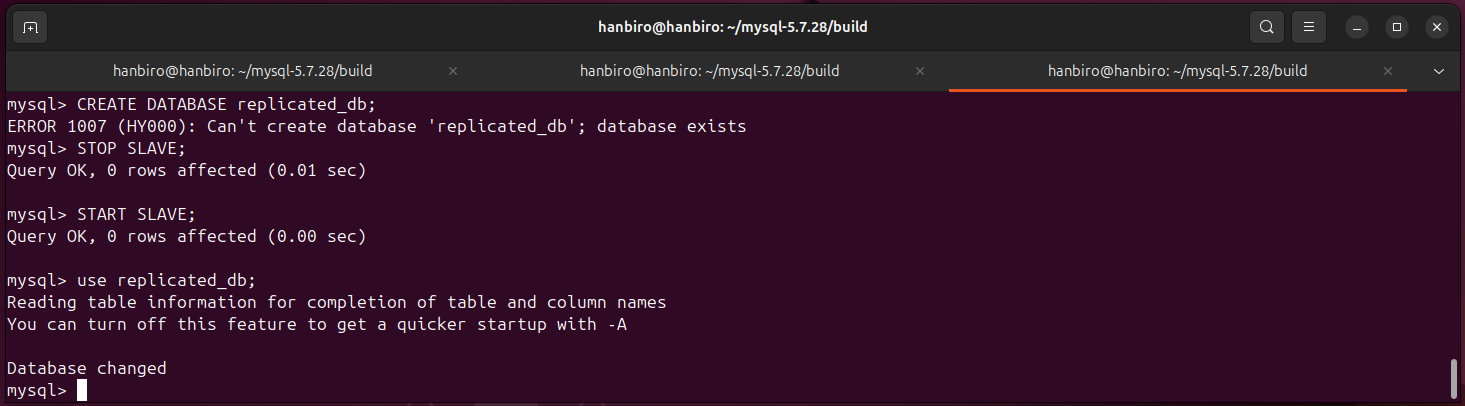
CREATE TABLE test\_table (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100)

);

**\*check; if at Slave hasn’t database, create database in Slave**

****

insert data into table on Master:

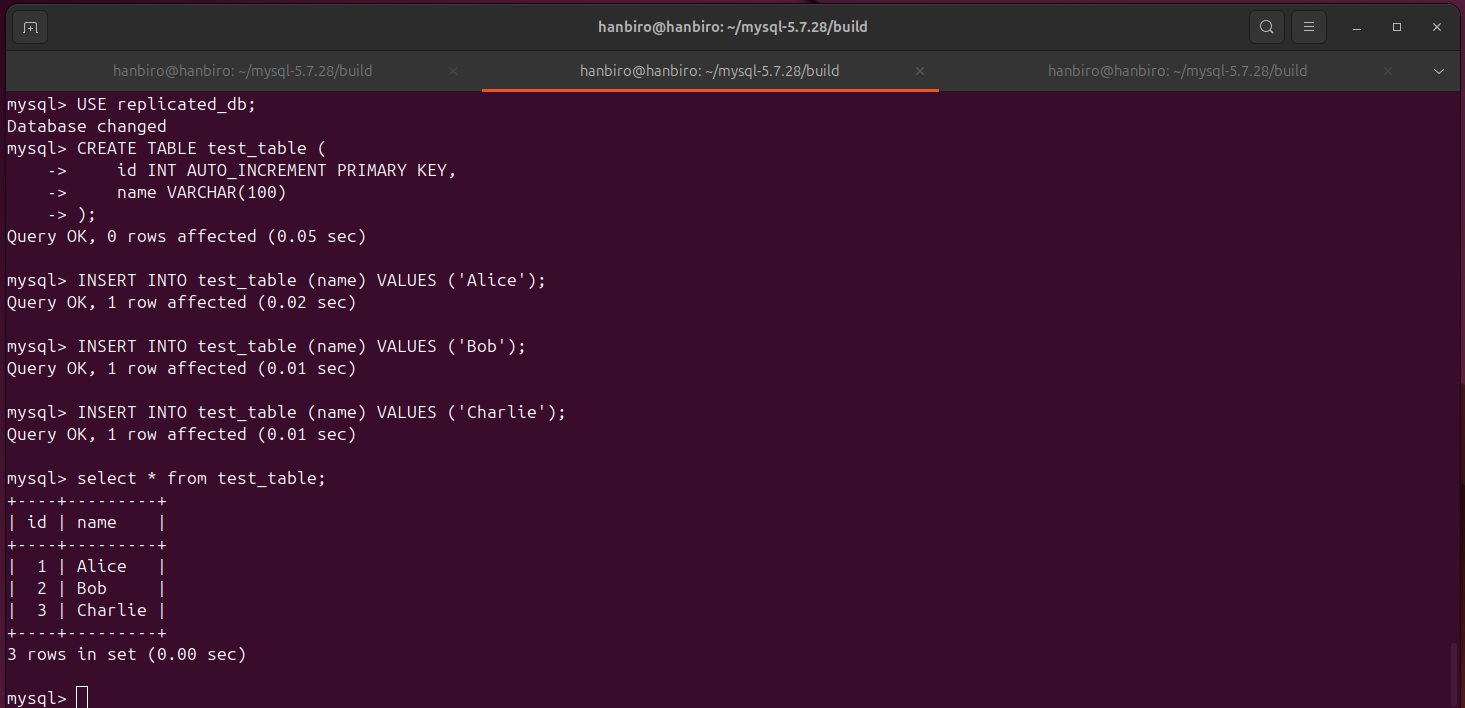
INSERT INTO test\_table (name) VALUES ('Alice');

INSERT INTO test\_table (name) VALUES ('Bob');

INSERT INTO test\_table (name) VALUES ('Charlie');

check data on Master:

SELECT \* FROM test\_table;



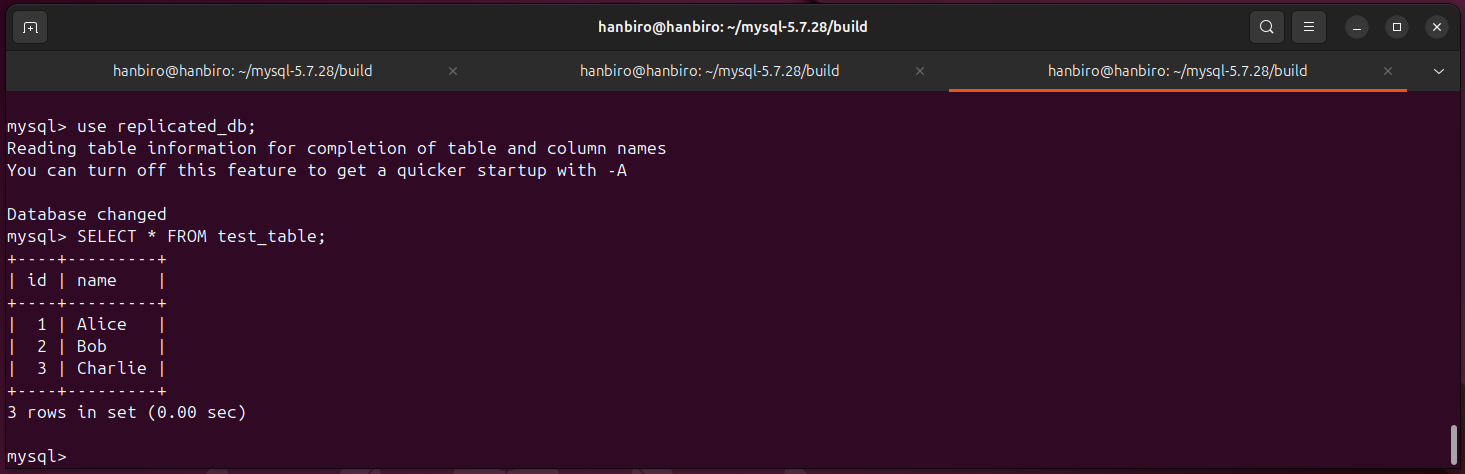
* **Check data on Slave:**

connect to Slave MySQL:

/usr/local/mysql/bin/mysql -uroot -S /tmp/mysql\_slave.sock

check data on Slave:

SELECT \* FROM test\_table;

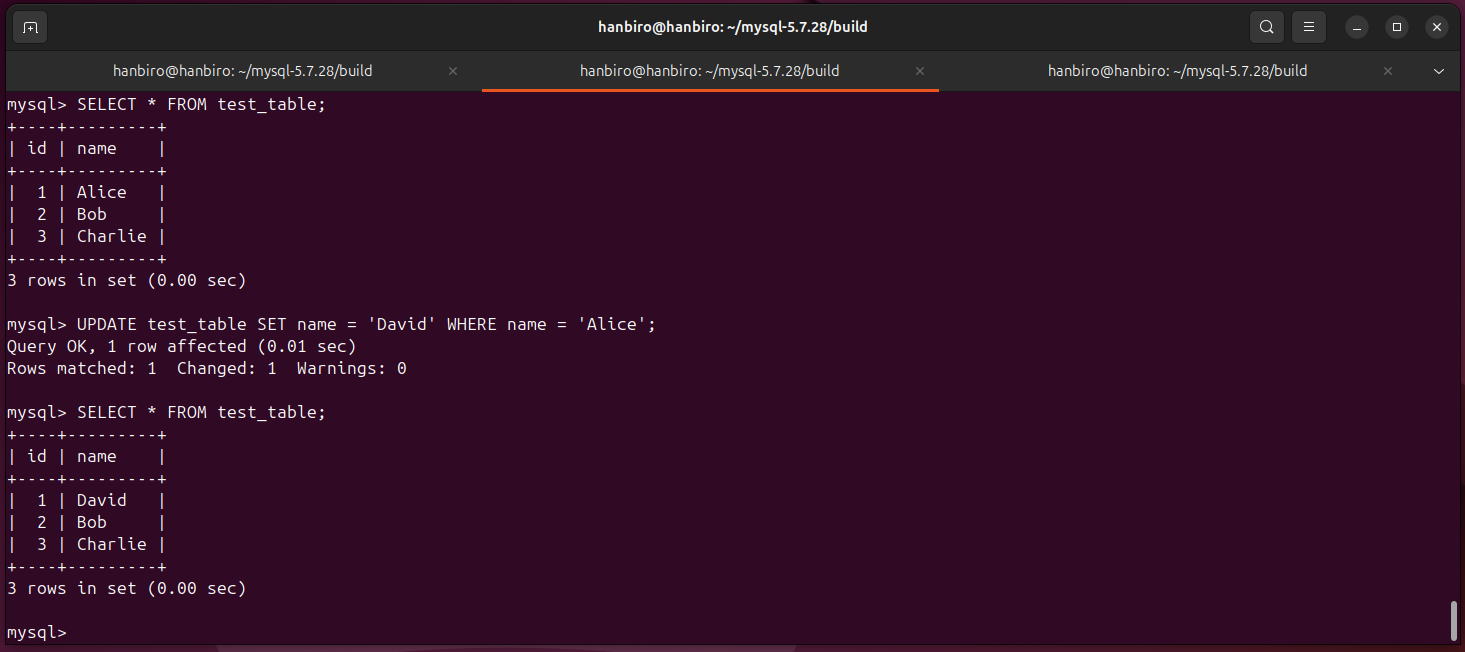


* **Check UPDATE:**

on Master:

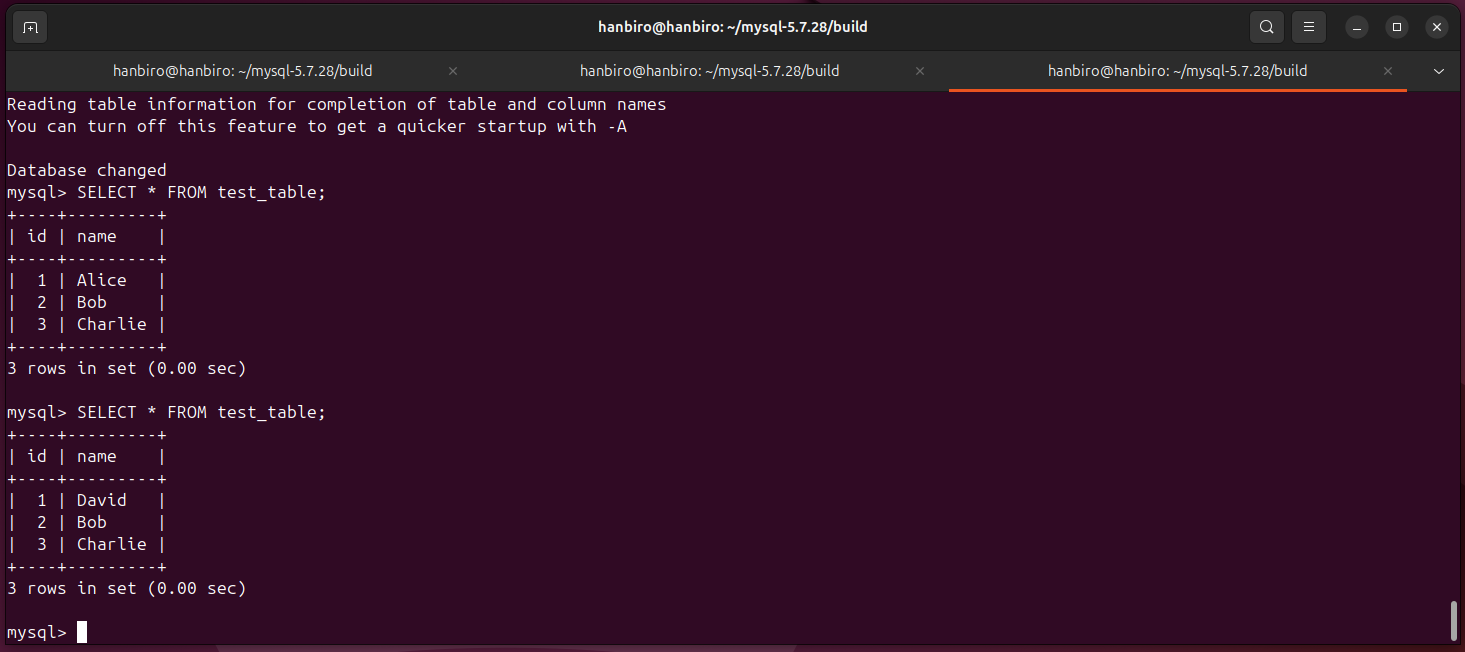
UPDATE test\_table SET name = 'David' WHERE name = 'Alice';

SELECT \* FROM test\_table;



on Slave:

SELECT \* FROM test\_table;

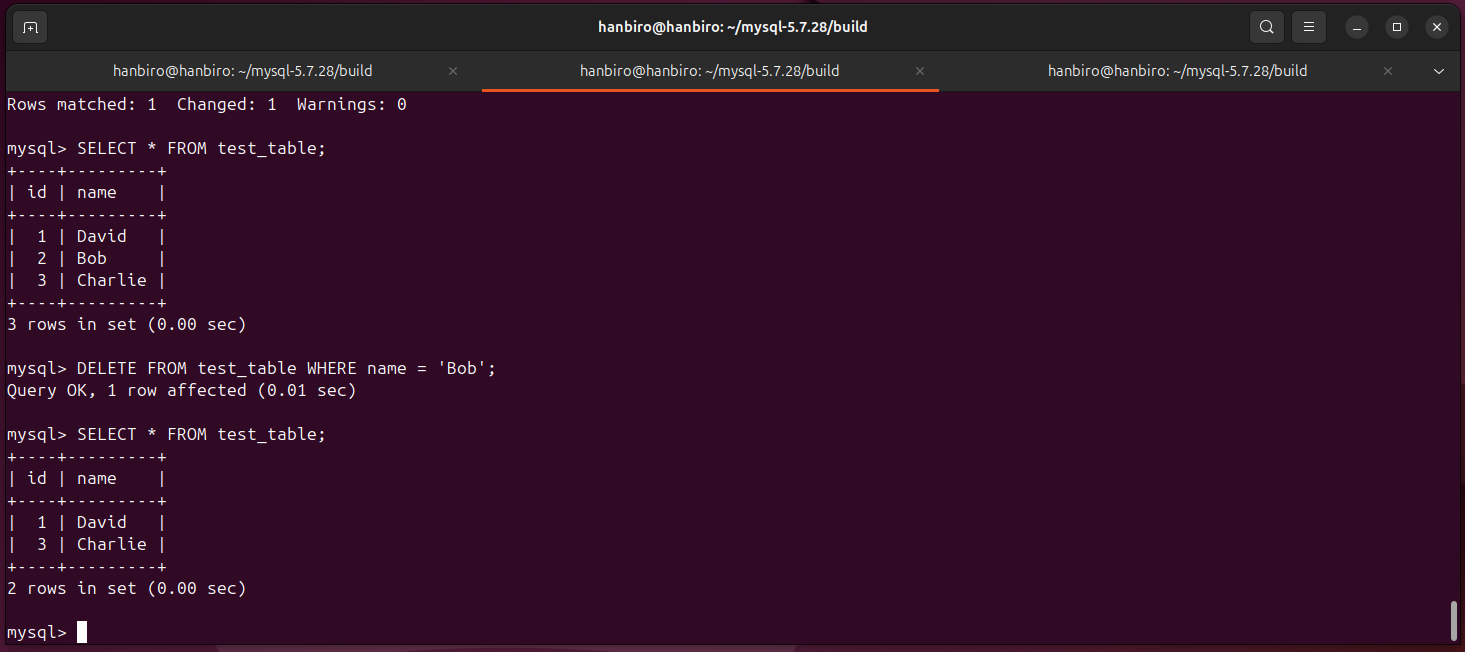


* **Check DELETE**

on Master:

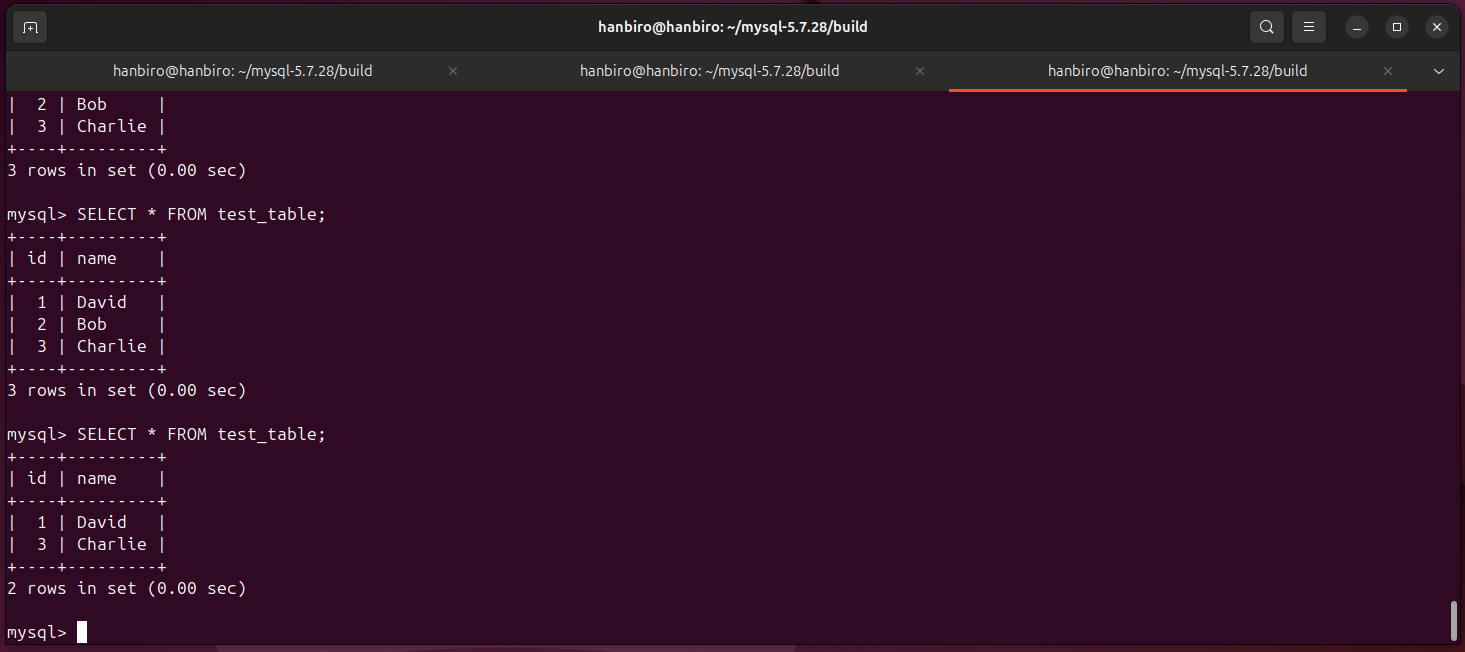
DELETE FROM test\_table WHERE name = 'Bob';

SELECT \* FROM test\_table;



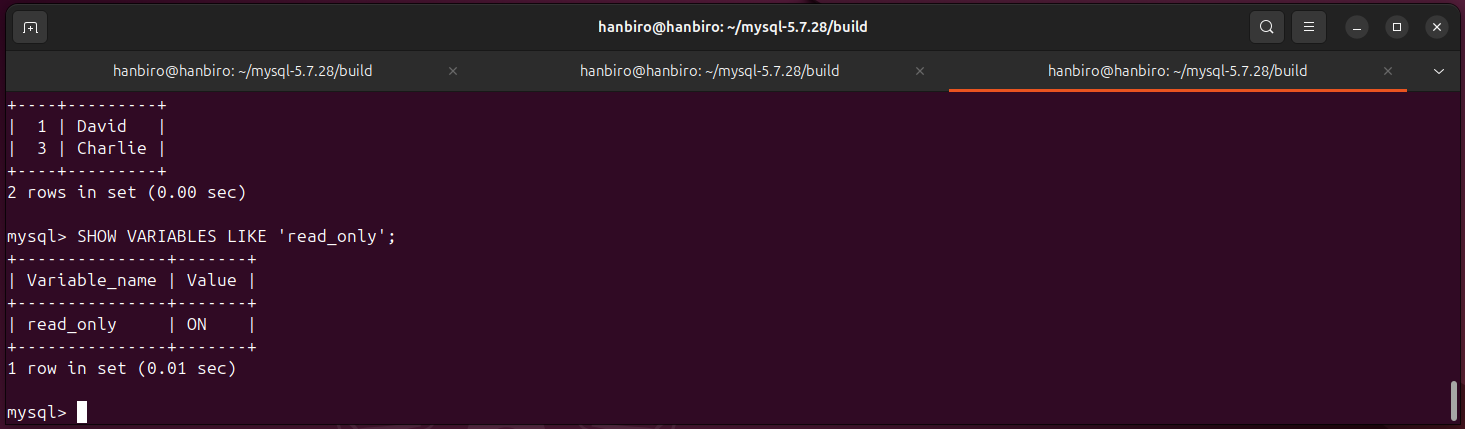
on Slave:

SELECT \* FROM test\_table;



**Configure read\_only=1 on Slave:**

When setting read\_only=1 on Slave, queries that change data will be stopped or ignored. This helps protect Slave from accidental or erroneous changes, since Slave can only be updated via replication from Master



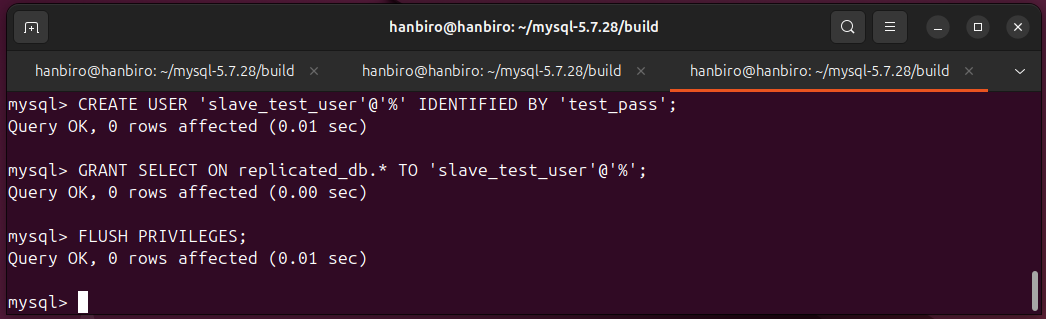
**[ Test with Normal User (Non-root User) ]**

* Create a **non-root user** on the **Slave**:

CREATE USER 'slave\_test\_user'@'%' IDENTIFIED BY 'test\_pass';

GRANT SELECT ON replicated\_db.\* TO 'slave\_test\_user'@'%';

FLUSH PRIVILEGES;



* Then, log in to Slave with normal user (create new Terminal):

/usr/local/mysql/bin/mysql -uslave\_test\_user -p -S /tmp/mysql\_slave.sock

*# Enter password: test\_pass*

* **Try performing the following CRUD and DDL operations on the Slave:**

**CREATE INSERT UPDATE DELETE ALTER DROP:**

CREATE DATABASE not\_allowed;

INSERT INTO replicated\_db.test\_table (name) VALUES ('TestFail');

UPDATE replicated\_db.test\_table SET name='Fail' WHERE id=1;

DELETE FROM replicated\_db.test\_table WHERE id=1;

ALTER TABLE replicated\_db.test\_table ADD COLUMN age INT;

DROP TABLE replicated\_db.test\_table;

**SELECT:** SELECT \* FROM replicated\_db.test\_table;

*\*result:*

***mysql> CREATE DATABASE not\_allowed;***

*ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement*

***mysql> INSERT INTO replicated\_db.test\_table (name) VALUES ('TestFail');***

*ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement*

***mysql> UPDATE replicated\_db.test\_table SET name='Fail' WHERE id=1;***

*ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement*

***mysql> DELETE FROM replicated\_db.test\_table WHERE id=1;***

*ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement*

***mysql> ALTER TABLE replicated\_db.test\_table ADD COLUMN age INT;***

*ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement*

***mysql> DROP TABLE replicated\_db.test\_table;***

*ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement*

***mysql> SELECT \* FROM replicated\_db.test\_table;***

*+----+---------+*

*| id | name |*

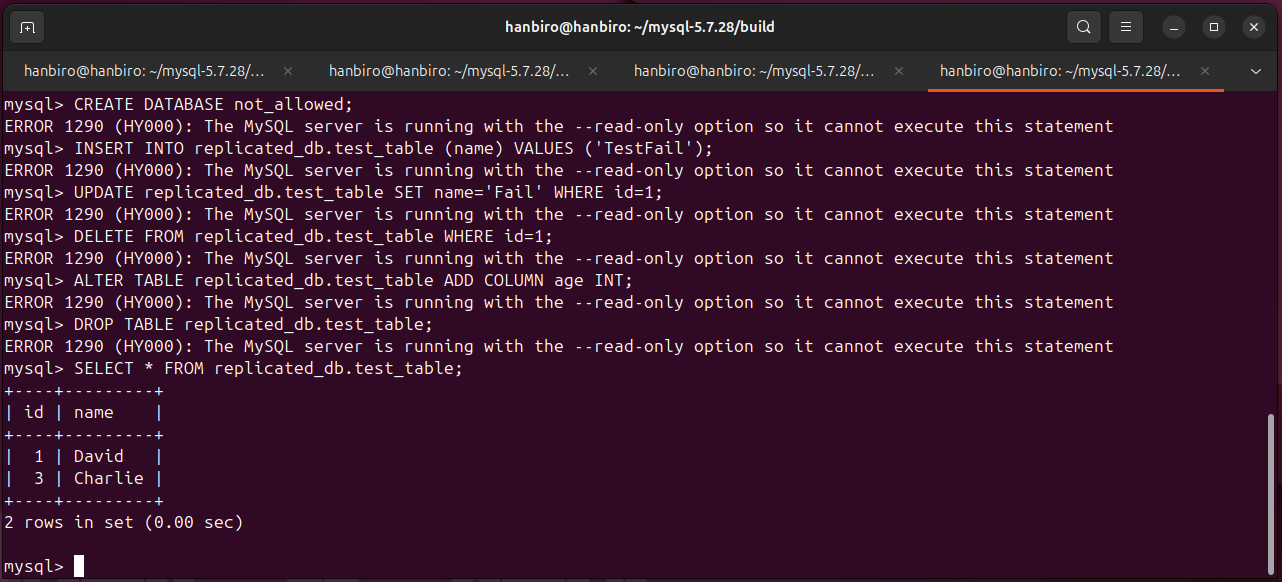
*+----+---------+*

*| 1 | David |*

*| 3 | Charlie |*

*+----+---------+*

*2 rows in set (0.00 sec)*



## **Does the data entered on the Master server applied to all to the Slave server? [ additional ]**

Yes, the data entered on the Master server will be automatically replicated to the Slave server. In a MySQL Master-Slave replication setup, the Master server records all changes (INSERT, UPDATE, DELETE) in its binary log. The Slave server then reads these logs and applies the changes to keep its data synchronized with the Master

This means that any data added, modified, or deleted on the Master server will be automatically replicated to the Slave server, ensuring consistency between them without the need for manual intervention. However, replication is typically one-way: changes on the Master are propagated to the Slave, but not vice versa unless specifically configured for multi-master replication

## **What are the required firewall settings?**

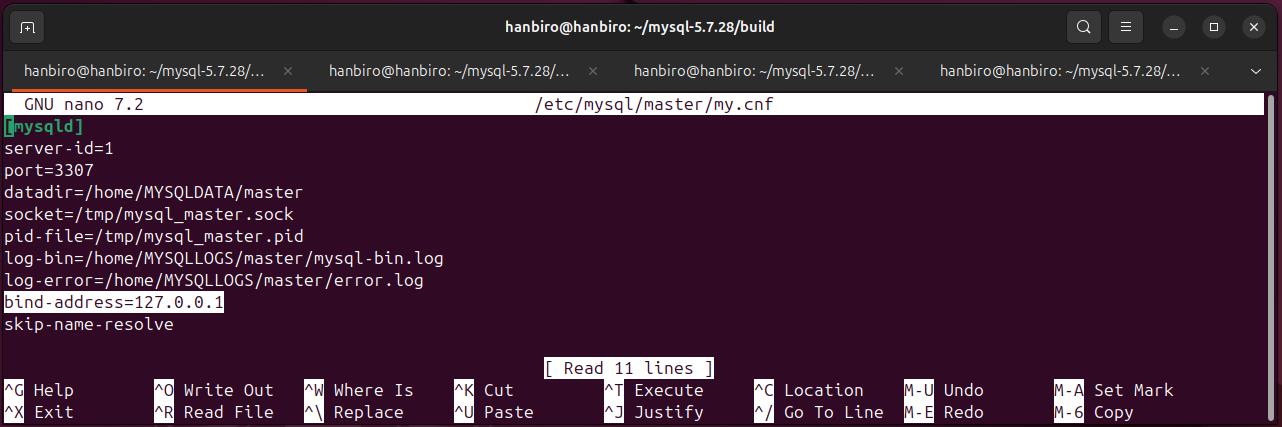
When deploying MySQL replication between Master and Slave, firewall is an important factor to ensure secure connection between servers, especially when deploying replication between separate physical servers (not running on the same machine)

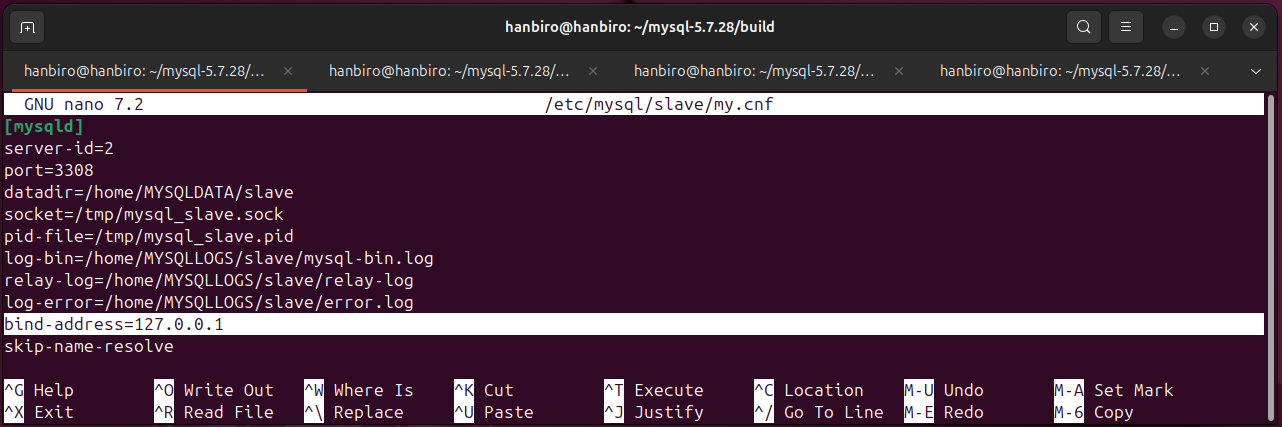
However, in case of running Master and Slave on the same machine (localhost), firewall is not necessary to configure because:

* Master and Slave communicate via localhost (127.0.0.1)
* The connection ports between Master and Slave (3307 and 3308) are only listened on localhost, so there is no connection from outside

Note:

Bind-address in MySQL: Make sure that the bind-address in my.cnf of both Master and Slave is configured as 127.0.0.1 (or 0.0.0.0 if remote access is needed in the future)

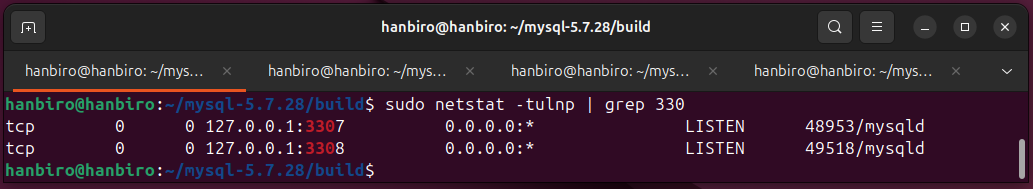




No need to open ports on the firewall as the connections between Master and Slave are internal

* Test internal connection between Master and Slave

Use netstat to check ports 3307 and 3308 on localhost: **sudo netstat -tulnp | grep 330**



## **What are the backup and restore procedures during operating?**

**Objectives**

* Protect data when errors occur (data loss, server failure, user error)
* Support quick recovery and resynchronization of replication when needed

**Backup Procedure**

* Backup on Master

Backup from Master to ensure accurate data, no replication delay

Use tools:

* **mysqldump** - simple, logical integrity (used for small-medium data)
* **mysqlhotcopy**, **mysqlpump** - replace or optimize mysqldump
* **Percona XtraBackup** - used when hot backup is needed without locking tables (large products)

**Backup on Slave**

Does not affect Master performance

Note: Need to stop replication (STOP SLAVE) temporarily to ensure data integrity in snapshot

**Restore Procedure**

Restore is usually performed on a new Master or recreates a replica if the Slave fails

After restore:

* Log position needs to be recorded or
* Re-establish replication using CHANGE MASTER TO

**[ TEST ]**

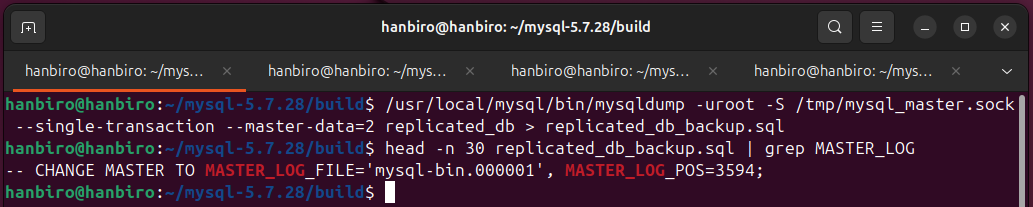
**Backup with mysqldump (on Master)**

* backup the entire **replicated\_db** database on Master

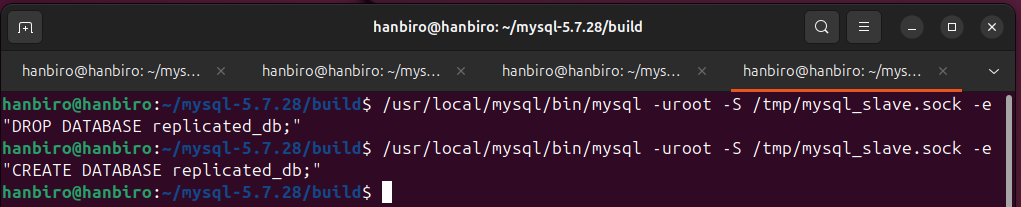
/usr/local/mysql/bin/mysqldump -uroot -S /tmp/mysql\_master.sock --single-transaction --master-data=2 replicated\_db > replicated\_db\_backup.sql

\*with: **--master-data=2** add binary log file and location information for replication recovery

**small problem:** when I sleep my computer, the File and Pos values ​​change



* Stop replication on Slave (purpose: Delete old data that is no longer correct, prepare for new data from backup)



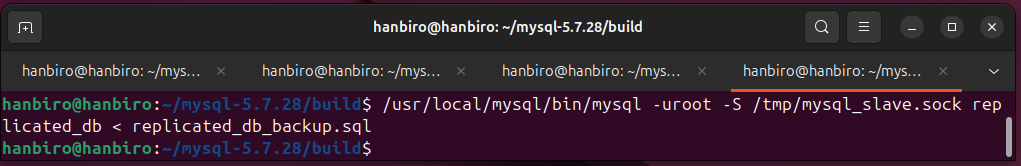
* Restore from backup file (

is in the directory containing the replicated\_db\_backup.sql file (~/mysql-5.7.28/build)

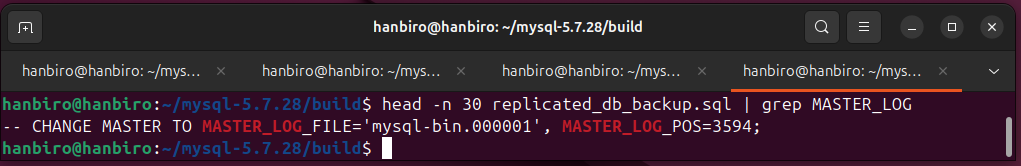
The command below will load the entire backup content into the Slave

)

/usr/local/mysql/bin/mysql -uroot -S /tmp/mysql\_slave.sock replicated\_db < replicated\_db\_backup.sql

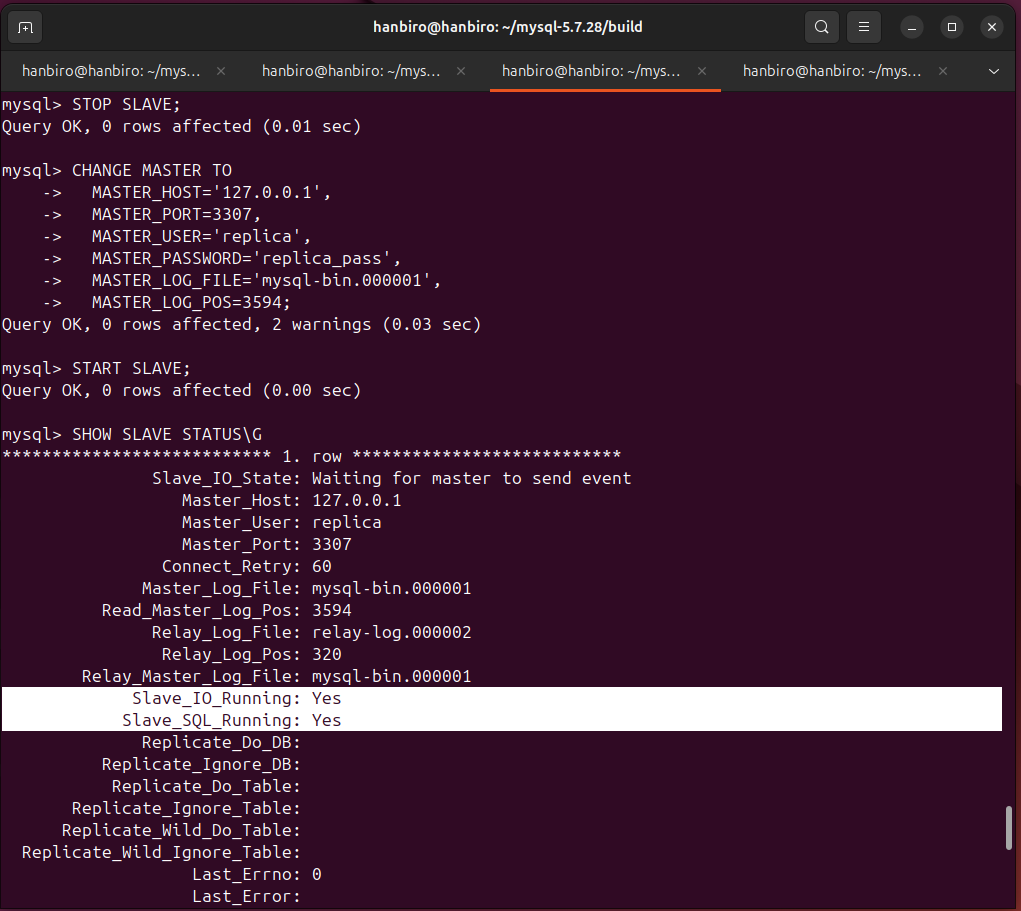


* Reset replication (with new log information)

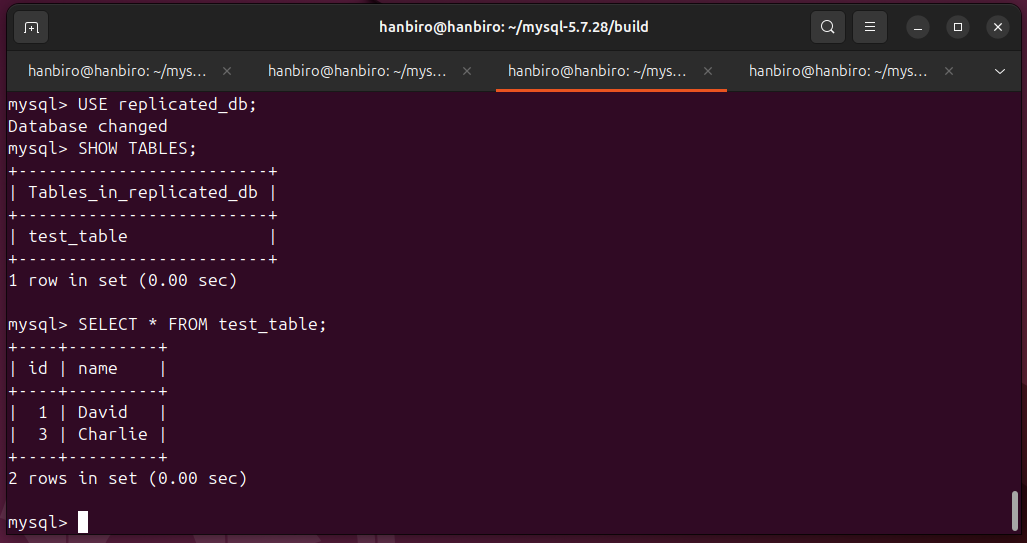


* connect to Slave

/usr/local/mysql/bin/mysql -uslave\_test\_user -p -S /tmp/mysql\_slave.sock



* check Database and Table on Slave:



**What are the pros of this configuration?**

**Architectural Overviews:**

**OS Ubuntu** (1 machine)

**MySQL 2 instance** MySQL 5.7.28, build from source

**Master port 3307**, **data** at /home/MYSQLDATA/**master**

**Slave port 3308**, **data** at /home/MYSQLDATA/**slave**

**Master-Slave replication** (log-bin / relay-log)

Query **read\_only=1** on **Slave**

Connection Using private sockets (/tmp/**mysql\_master.sock** and /tmp/**mysql\_slave.sock**)

**Replica** user for **replication**, **slave\_test\_user** only **SELECT**

\*WITH:

**Separation of Write and Read Operations**

* The Master handles all data modifications (INSERT, UPDATE, DELETE)
* The Slave is configured as read\_only=1 and only handles SELECT queries
* This improves performance by offloading read-heavy operations to the Slave

**Improved Performance with Separate Storage Paths**

* Master and Slave have dedicated directories for data and logs:
* Data: **/home/MYSQLDATA/master** and **/home/MYSQLDATA/slave**
* Logs: **/home/MYSQLLOGS/master** and **/home/MYSQLLOGS/slave**
* This separation helps reduce disk I/O contention

**Independent Configuration and Port Assignment**

* Each MySQL instance uses a custom configuration file:
* Master: /etc/mysql/**master/my.cnf**, port **3307**
* Slave: /etc/mysql/**slave/my.cnf**, port **3308**
* Custom socket files prevent conflicts:
* **/tmp/mysql\_master.sock** and **/tmp/mysql\_slave.sock**

**Safe and Uninterrupted Backup**

* Backups can be executed on the **Slave** using **mysqldump**
* This avoids interrupting the **Master**, ensuring data consistency with **--single-transaction**

**Safe Environment for Testing and Monitoring**

* Slave can be used to run reports or test SELECT queries without impacting production data
* Prevents accidental data loss due to enforced read-only mode

**Easy to Scale**

* The current architecture allows to add more Slaves if needed
* The system is prepared for distributed setups

**What are the cons of this configuration?**

**Replication Delay (Lag)**

* Data written to the Master may not appear immediately on the Slave
* Causes issues for applications that rely on real-time reads after write

**No Automatic Failover**

* If the Master goes down, the system does not switch automatically to the Slave
* Manual promotion of Slave is required and may lead to downtime

**Slave is Read-Only**

* Cannot perform any write operations or schema changes on the Slave
* Limits flexibility for testing write-based queries or applying emergency fixes

**Manual Setup and Maintenance Complexity**

* The configuration uses manually compiled MySQL from source
* Requires careful handling of socket files, custom ports, directories, and startup routines

**No Built-in Conflict Handling**

* If dual-master replication is introduced later without careful control, it may cause data conflicts
* Current setup does not support multi-source or circular replication

**Single Point of Write**

* Only one Master exists, making it a performance bottleneck for high write loads
* Requires upgrading to dual-master or sharding if write throughput increases

**Backup Risk If Slave Is Not Synchronized**

* If replication fails silently, backup taken from Slave may be outdated or inconsistent
* Requires regular monitoring of SHOW SLAVE STATUS

**What are the precautions for usage?**

**Configurations Already Implemented**

* Slave is set to read-only mode (read\_only=1)
* this prevents direct write operations on the Slave, ensuring data is only updated through replication from the Master
* Separate socket and port for each instance
* Master: port 3307, socket /tmp/mysql\_master.sock
* Slave: port 3308, socket /tmp/mysql\_slave.sock
* Avoids conflicts when running multiple instances on the same machine
* User created with SELECT-only permissions on Slave
* The user slave\_test\_user can only read data (SELECT), preventing any write or structural changes on the Slave
* Safe backup taken from Master
* Using mysqldump with --single-transaction --master-data=2 to ensure accurate binary log position for recovery
* Manual checking of replication status
* Using SHOW SLAVE STATUS\G to verify the replication process is working as expected

**Which Service is Suitable with this Configuration?**

**Suitable Services**

* **Read-Heavy Applications**

The Master-Slave setup allows distributing read queries to the Slave, offloading read-heavy traffic from the Master

ex: reporting systems, analytics platforms

* **Data Backup and Disaster Recovery**

The Slave can act as a backup and can be promoted to Master in case of failure, minimizing downtime

ex: critical systems requiring high availability

* **Business Intelligence and Reporting**

With read-only access on the Slave, BI tools can safely query without impacting the Master

ex: dashboards, reporting tools

**Not Suitable For:**

* **Real-Time Data Processing:** replication delay could affect real-time consistency, which is not suitable for time-sensitive applications
* **Write-Heavy Applications:** all writes go to the Master, which could cause a bottleneck in environments with high write frequency
* **Complex Transactions:** asynchronous replication can lead to temporary inconsistencies between Master and Slave, which may not be ideal for complex transactional systems

**Although there will be any problem during the test, how can you solve this problem when the log filses will keep increasing with the actual service ?**

* When replication is running, MySQL will continuously write to:
* On Master: binary log (mysql-bin.00000x)
* On Slave: relay log (relay-log.00000x)
* These files increase continuously when more data is written or replication is running for a long time.
* If not handled:
* Waste disk space
* Cause service interruption or stop

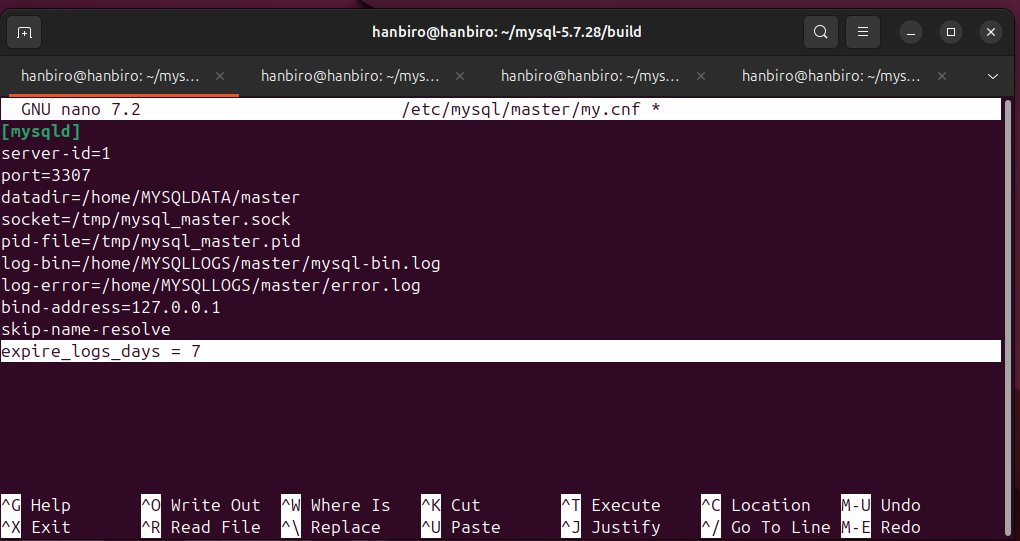
**Solution with my configuration**

Add to my.cnf file configuration:

* on Master: **sudo nano /etc/mysql/master/my.cnf**

**expire\_logs\_days = 7**

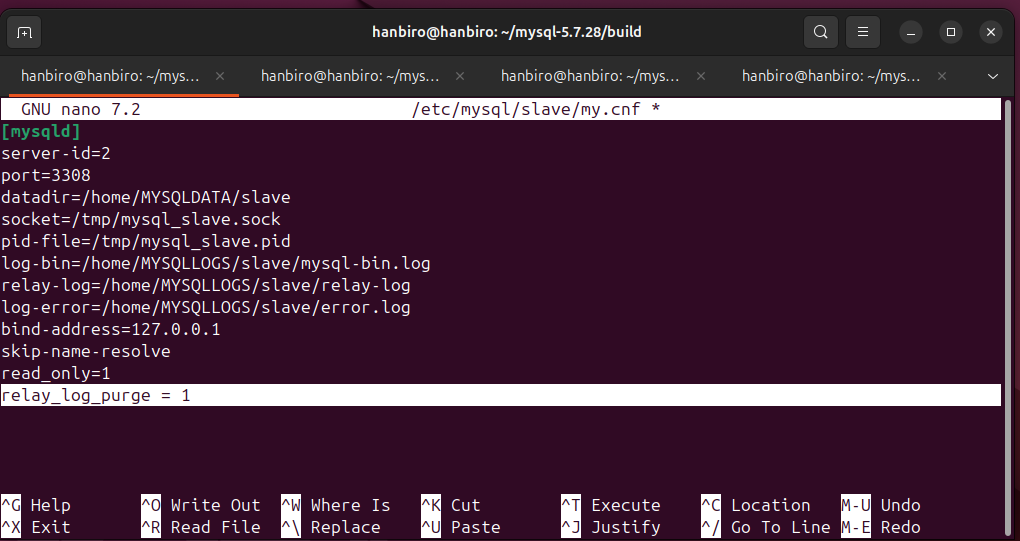
-> automatically delete binary log after 7 days



* on Slave: **sudo nano /etc/mysql/slave/my.cnf**

**relay\_log\_purge = 1**

-> MySQL will automatically delete the relay log when it is no longer needed



**Convert MySQL Replication to Dual Master**

**#** Create **replica** user on **Slave (port 3308)**

connect to Slave:

/usr/local/mysql/bin/mysql -uroot -S /tmp/mysql\_slave.sock

create reverse replication user:

CREATE USER 'replica'@'%' IDENTIFIED BY 'replica\_pass';

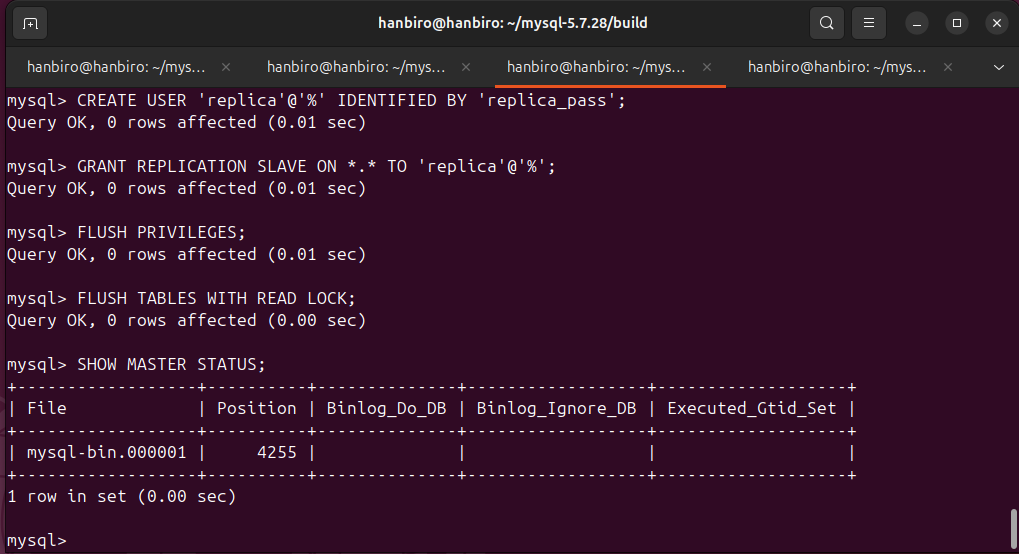
GRANT REPLICATION SLAVE ON \*.\* TO 'replica'@'%';

FLUSH PRIVILEGES;

FLUSH TABLES WITH READ LOCK;

SHOW MASTER STATUS;

\*with: File (mysql-bin.000001) and Position (4255)



**#** Set node **3307** as **Slave** **of** **3308**

* connect to Master:

/usr/local/mysql/bin/mysql -uroot -S /tmp/mysql\_master.sock

* Configure replication from 3308 to 3307:

STOP SLAVE;

CHANGE MASTER TO

MASTER\_HOST='127.0.0.1',

MASTER\_PORT=3308,

MASTER\_USER='replica',

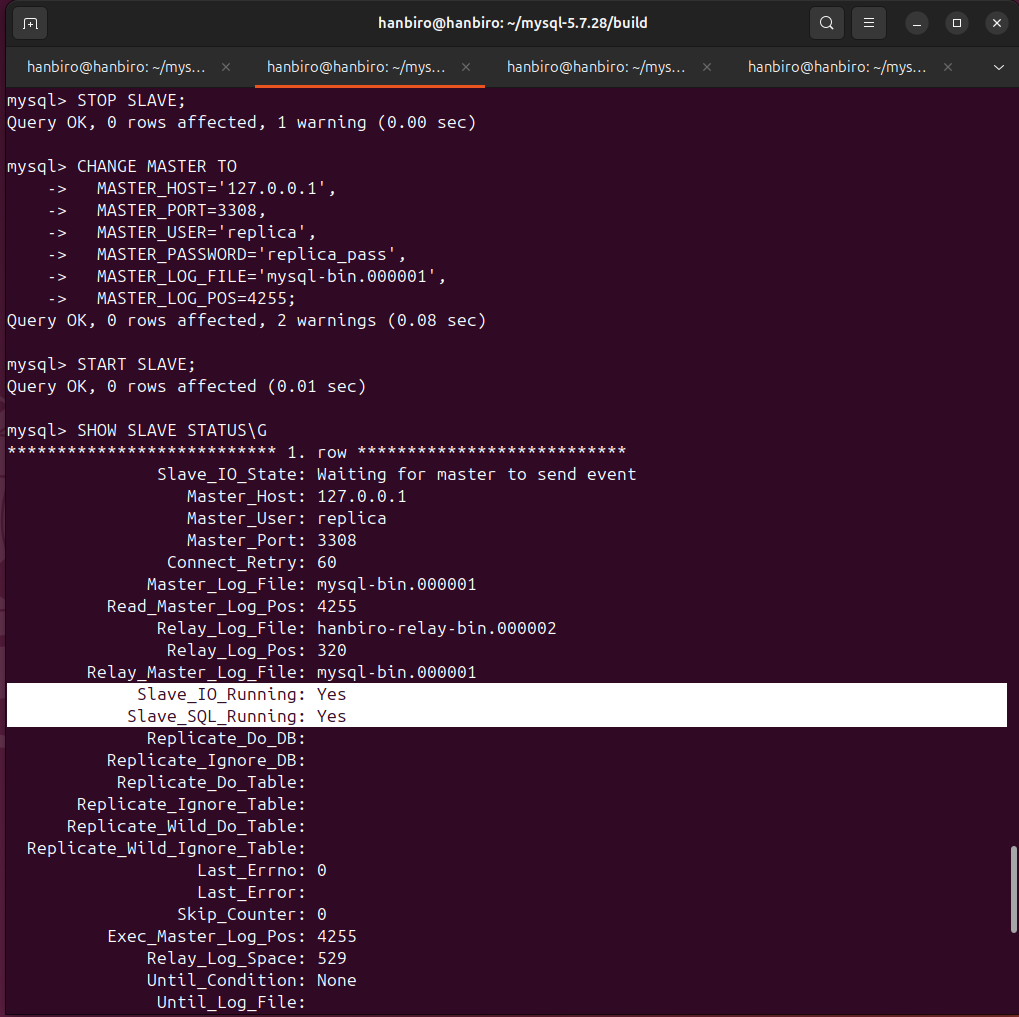
MASTER\_PASSWORD='replica\_pass',

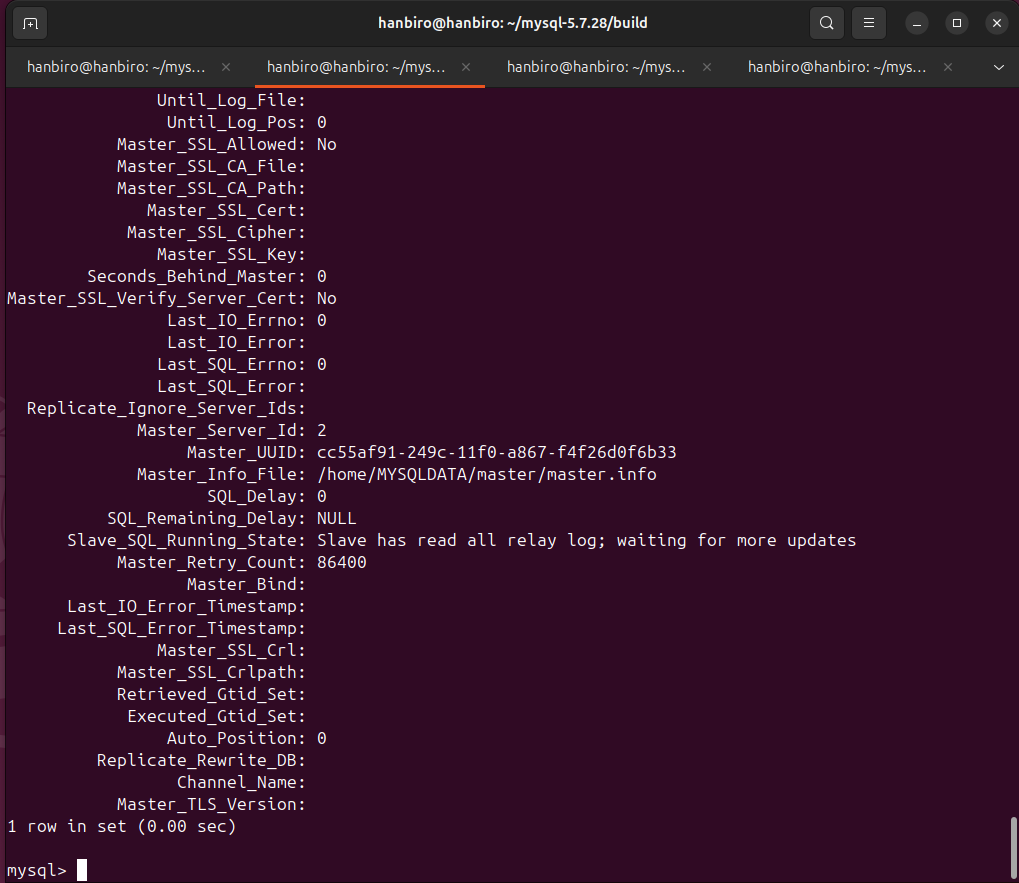
MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=4255;

START SLAVE;

SHOW SLAVE STATUS\G



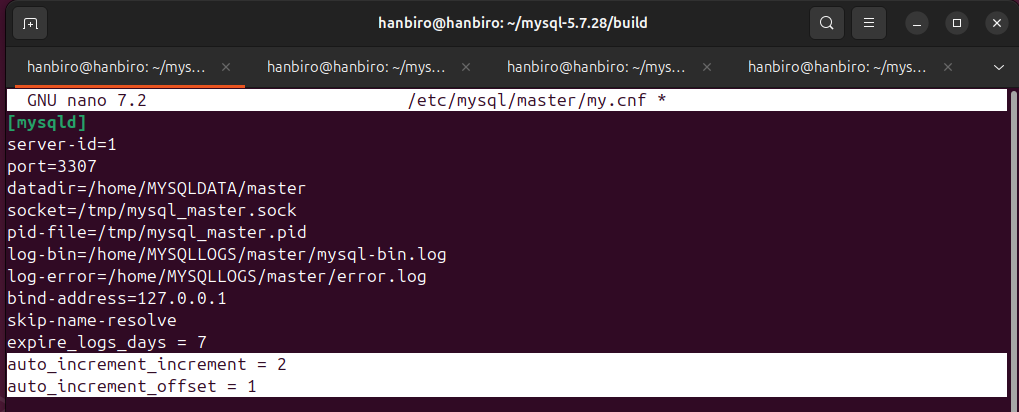


**#** Required notes to avoid **loop replication** or **duplicate primary key errors** (this prevents auto-incrementing IDs from being duplicated when written to both nodes)

* Using **auto\_increment\_increment** and **auto\_increment\_offset**
* on Master, **sudo nano /etc/mysql/master/my.cnf**

auto\_increment\_increment = 2

auto\_increment\_offset = 1

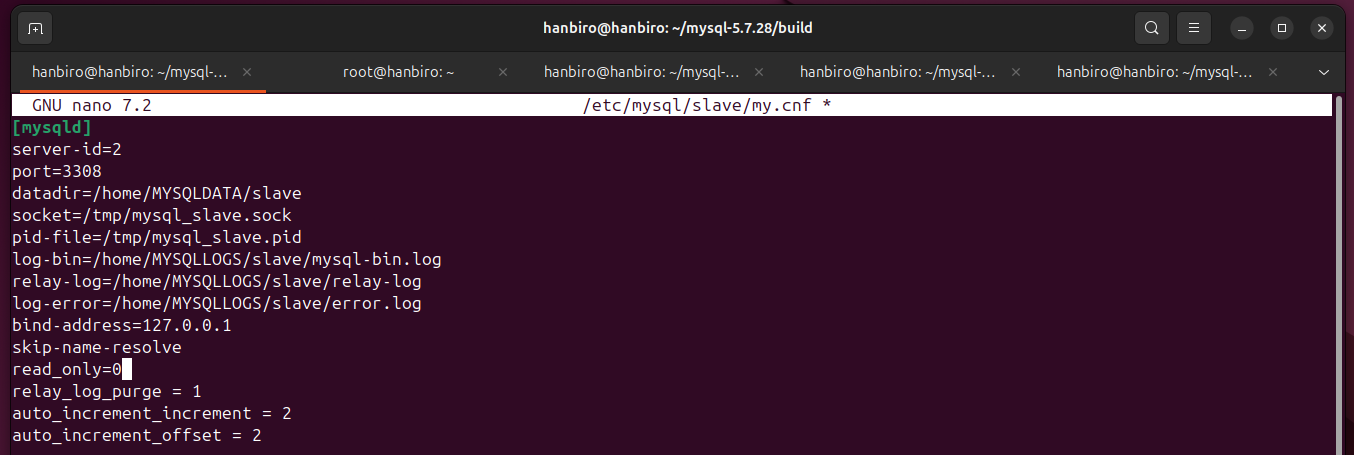


* on Slave, **sudo nano /etc/mysql/slave/my.cnf**

**read\_only=0**

auto\_increment\_increment = 2

auto\_increment\_offset = 2



* After configuration, restart both instances:

sudo pkill mysqld

sudo /usr/local/mysql/bin/mysqld\_safe --defaults-file=/etc/mysql/master/my.cnf &

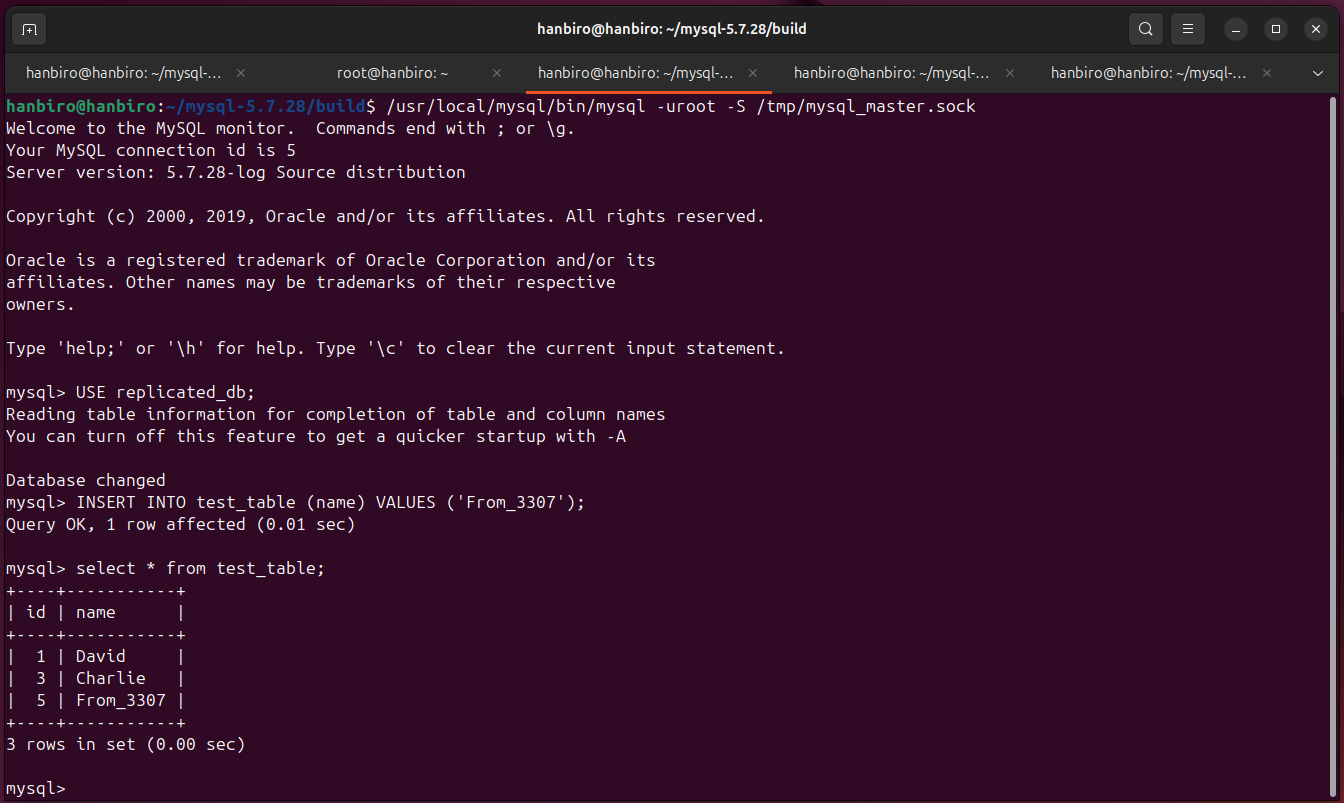
sudo /usr/local/mysql/bin/mysqld\_safe --defaults-file=/etc/mysql/slave/my.cnf &

**Test INSERT from Master 3307 → Slave 3308**

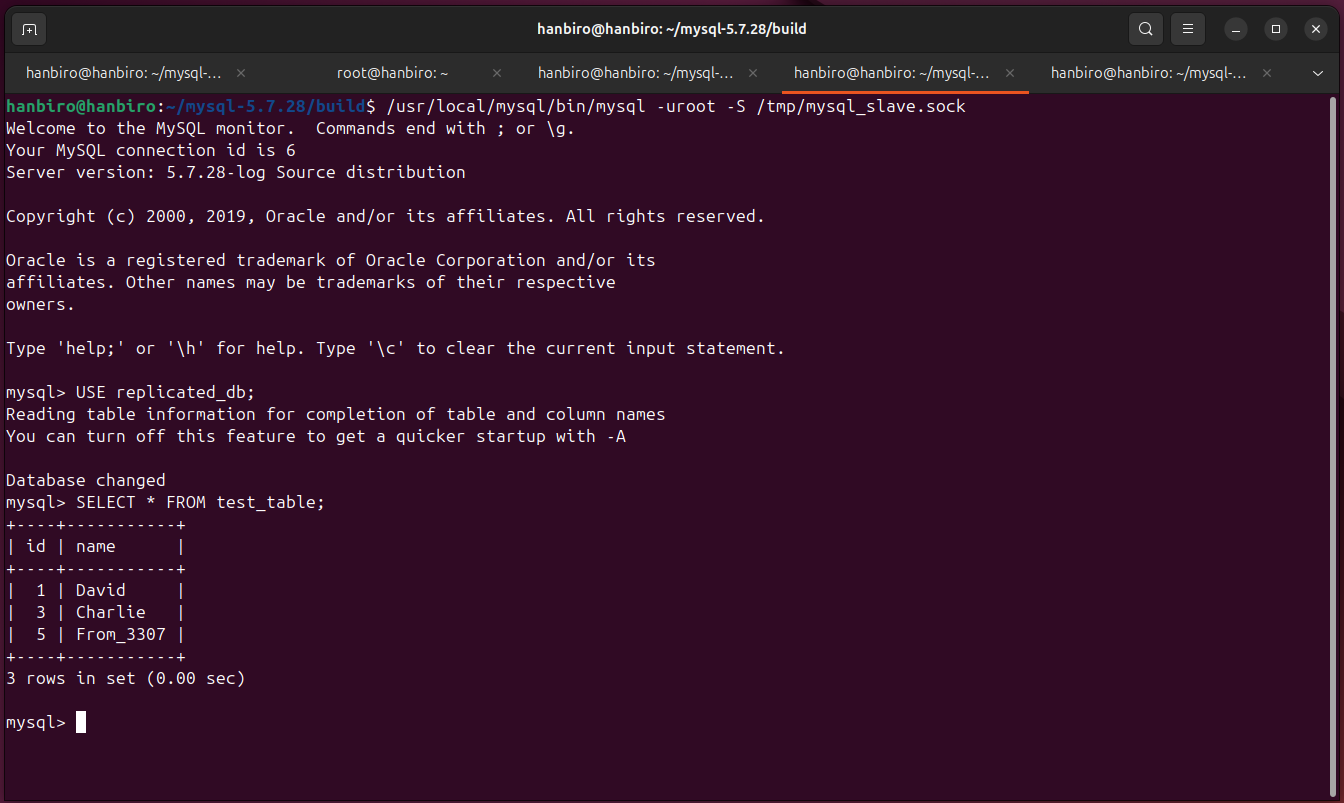
on **Master**:

USE replicated\_db;

INSERT INTO test\_table (name) VALUES ('From\_3307');



check on **Slave**:

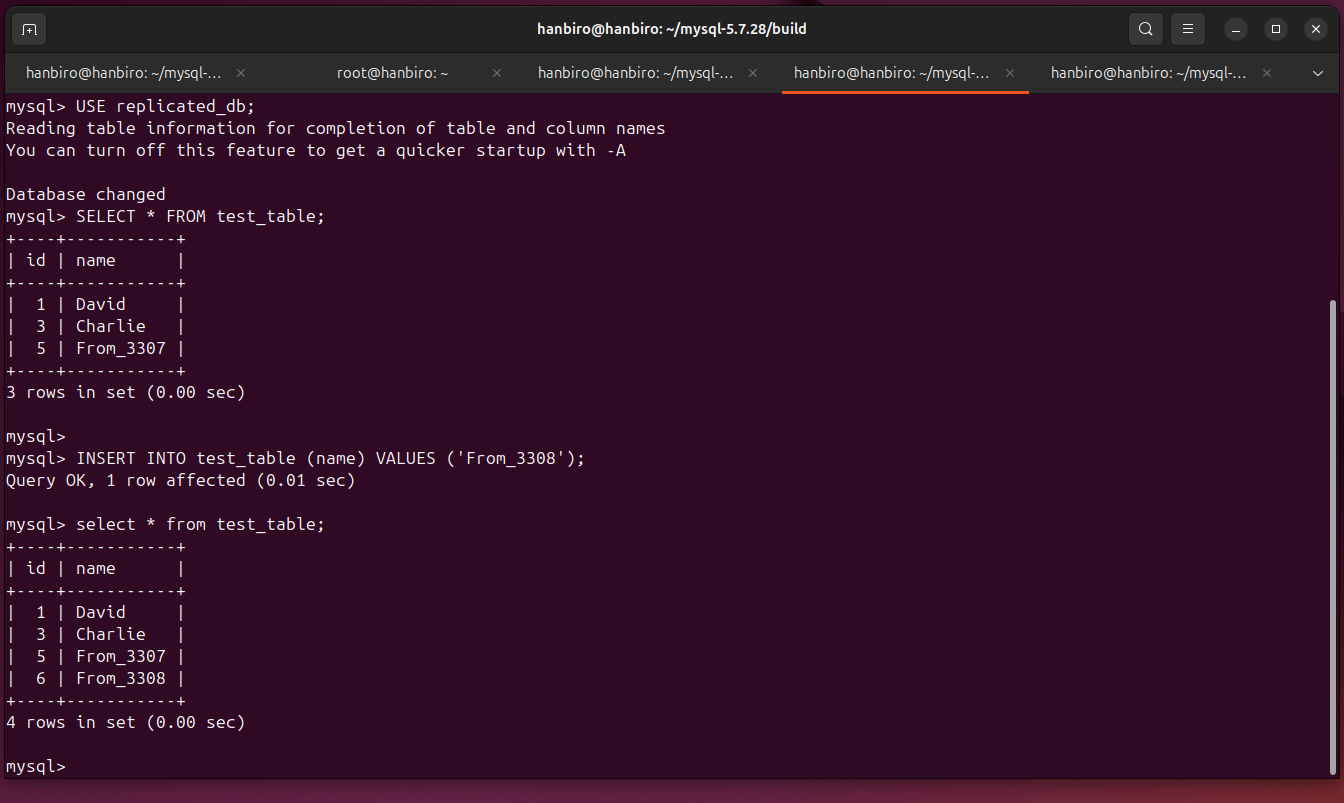


**Test INSERT from Slave 3308 → Master 3307**

on **Slave**:

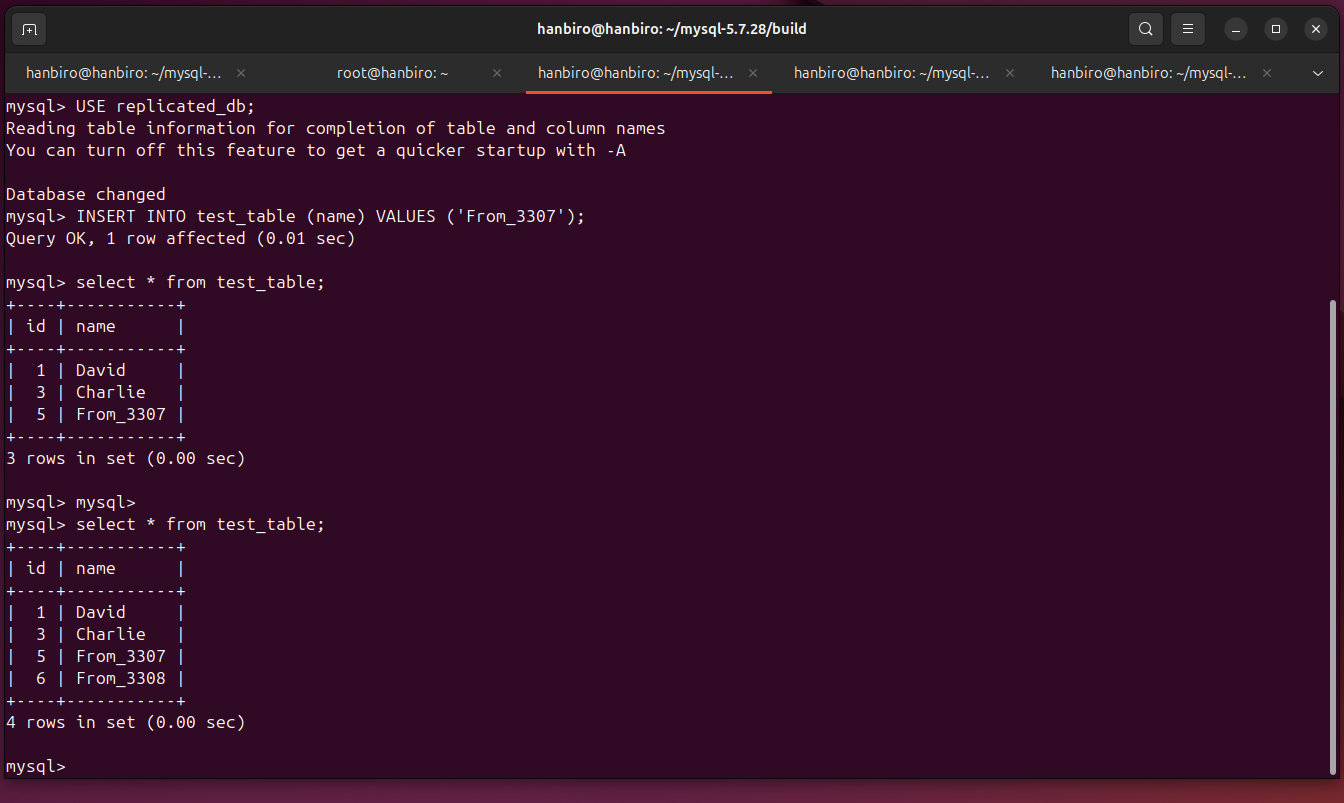
USE replicated\_db;

INSERT INTO test\_table (name) VALUES ('From\_3308');



check on **Master**:

select \* from test\_table;



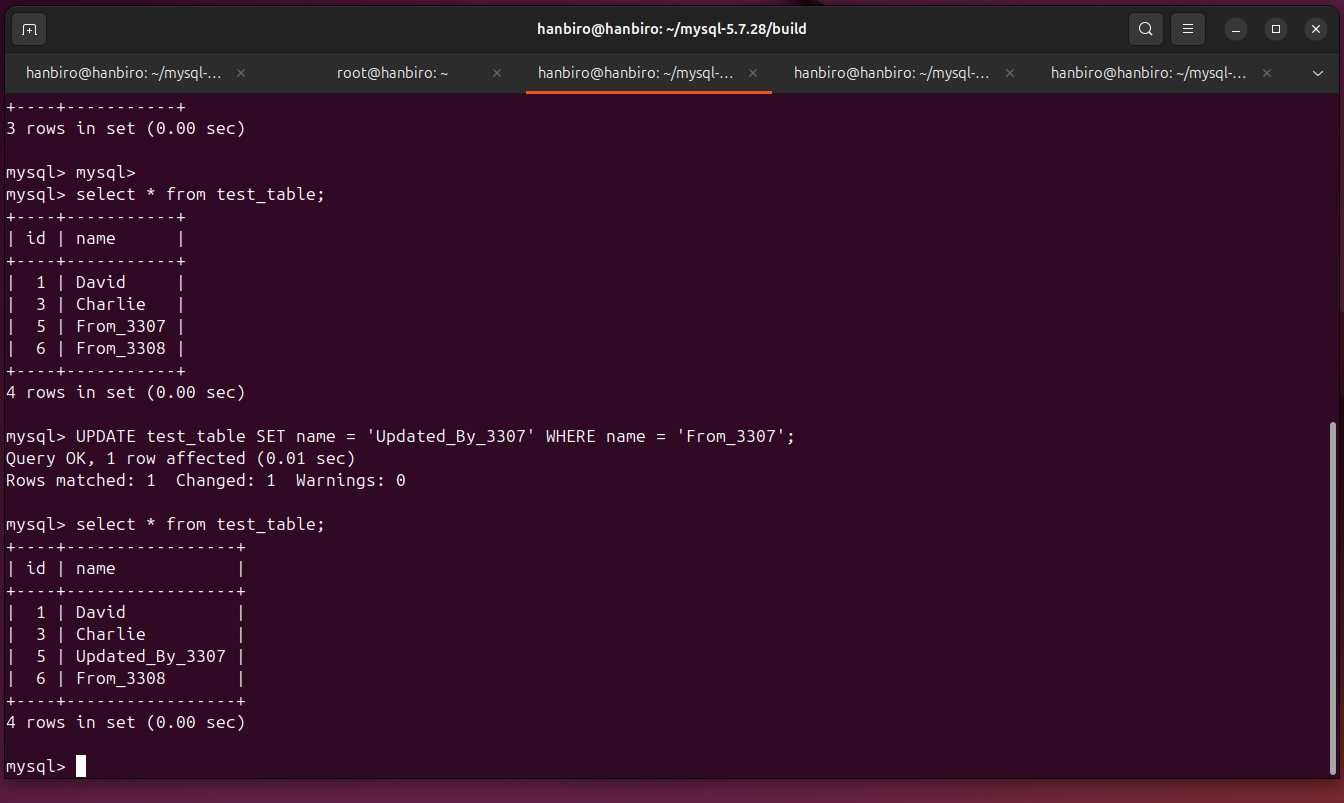
**2-way UPDATE Test**

**###** Test Update from **Master -** check on **Slave:**

* on **Master**:

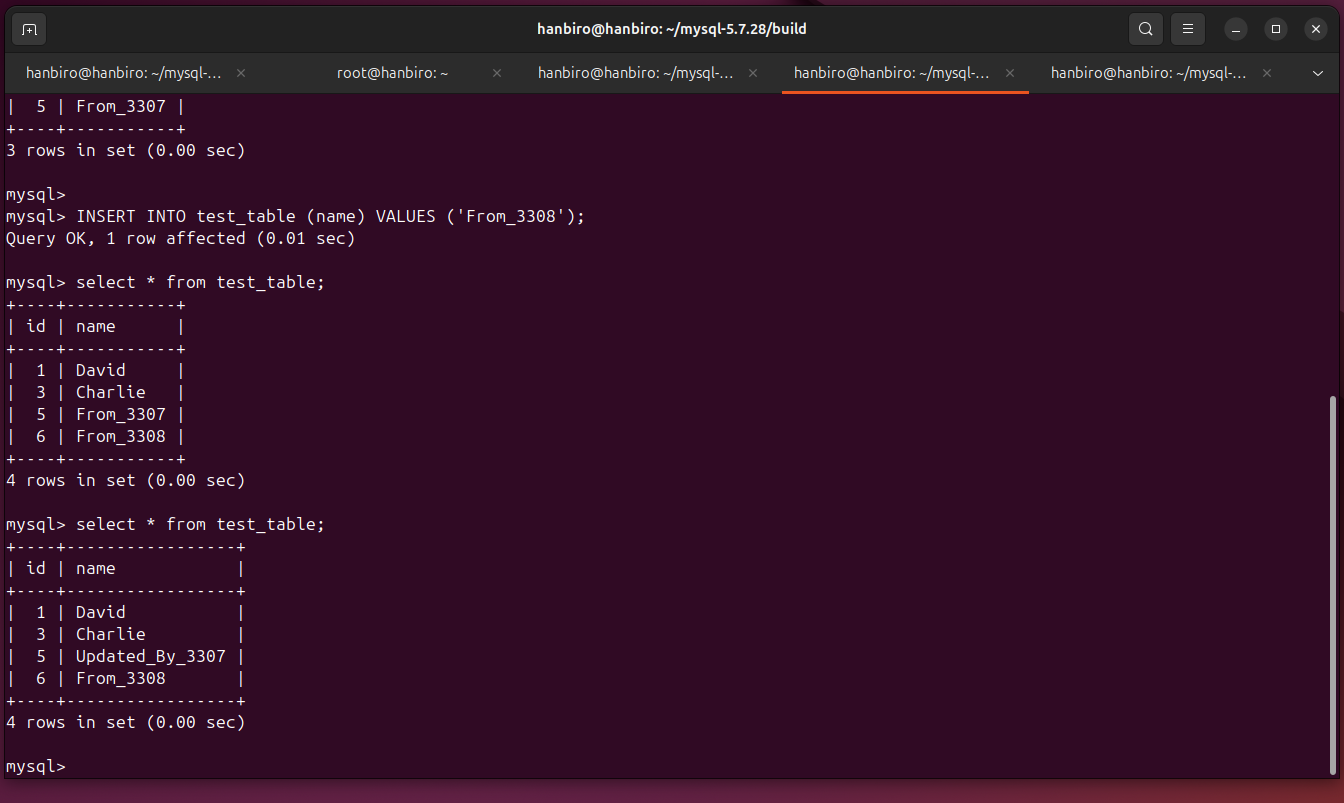
UPDATE test\_table SET name = 'Updated\_By\_3307' WHERE name = 'From\_3307';

select \* from test\_table;



* check on **Slave**:

select \* from test\_table;

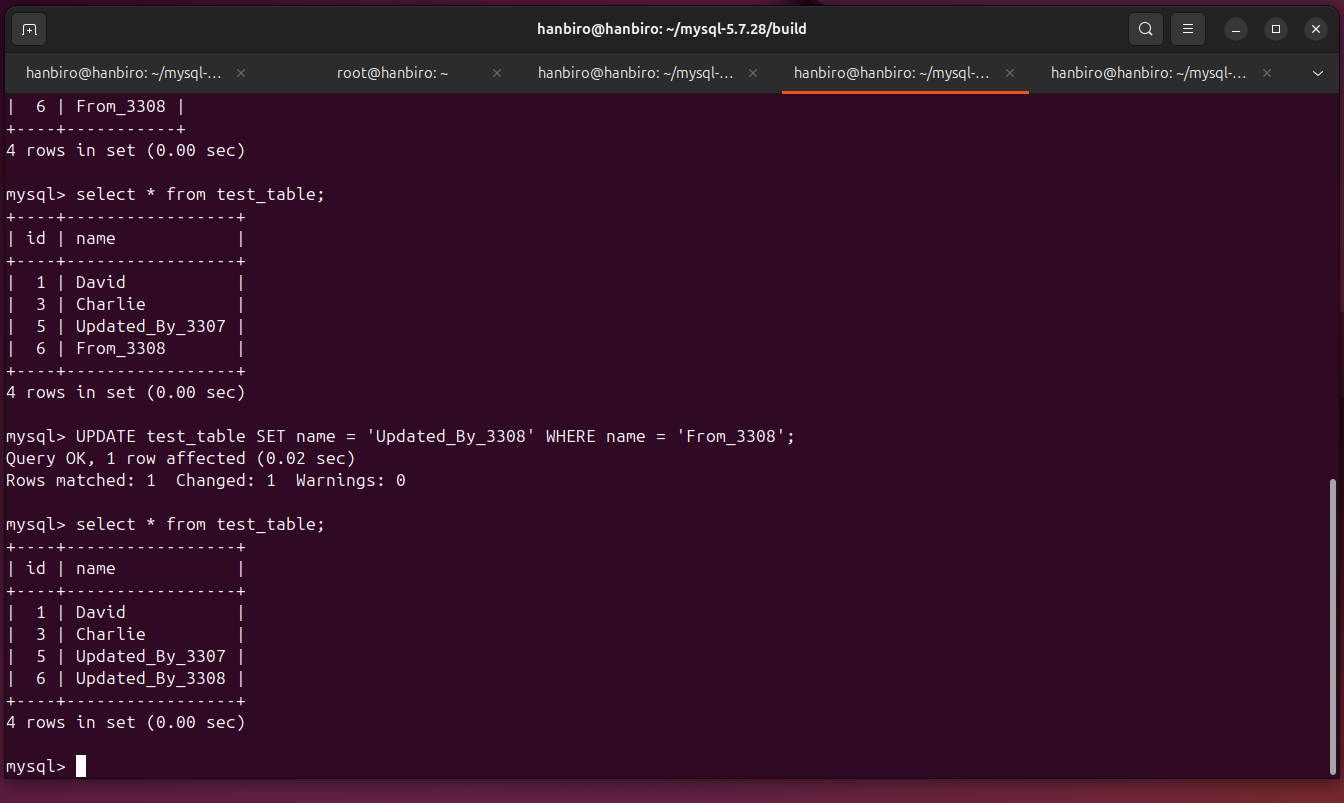


**###** Test Update from **Slave -** check on **Master:**

* on **Slave**:

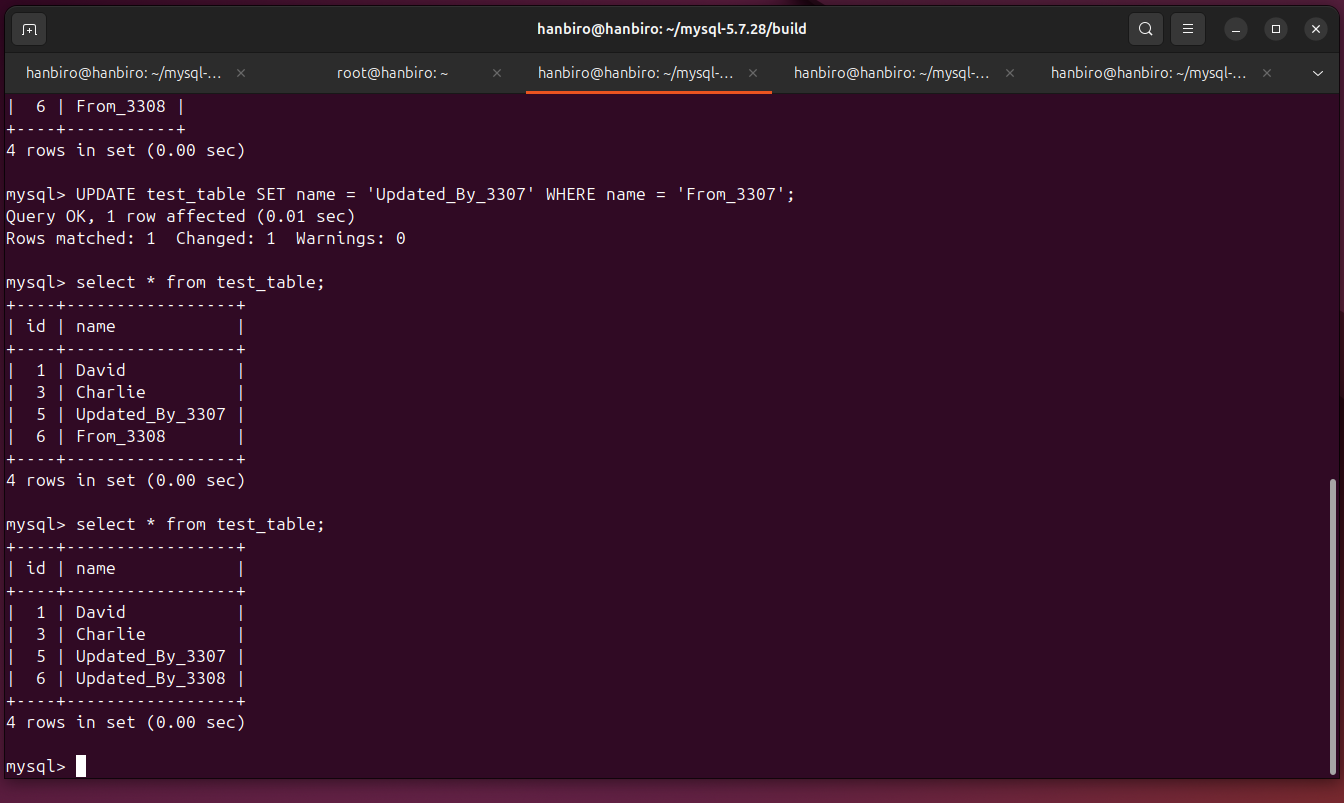
UPDATE test\_table SET name = 'Updated\_By\_3308' WHERE name = 'From\_3308';

select \* from test\_table;



* check on **Master**:

select \* from test\_table;



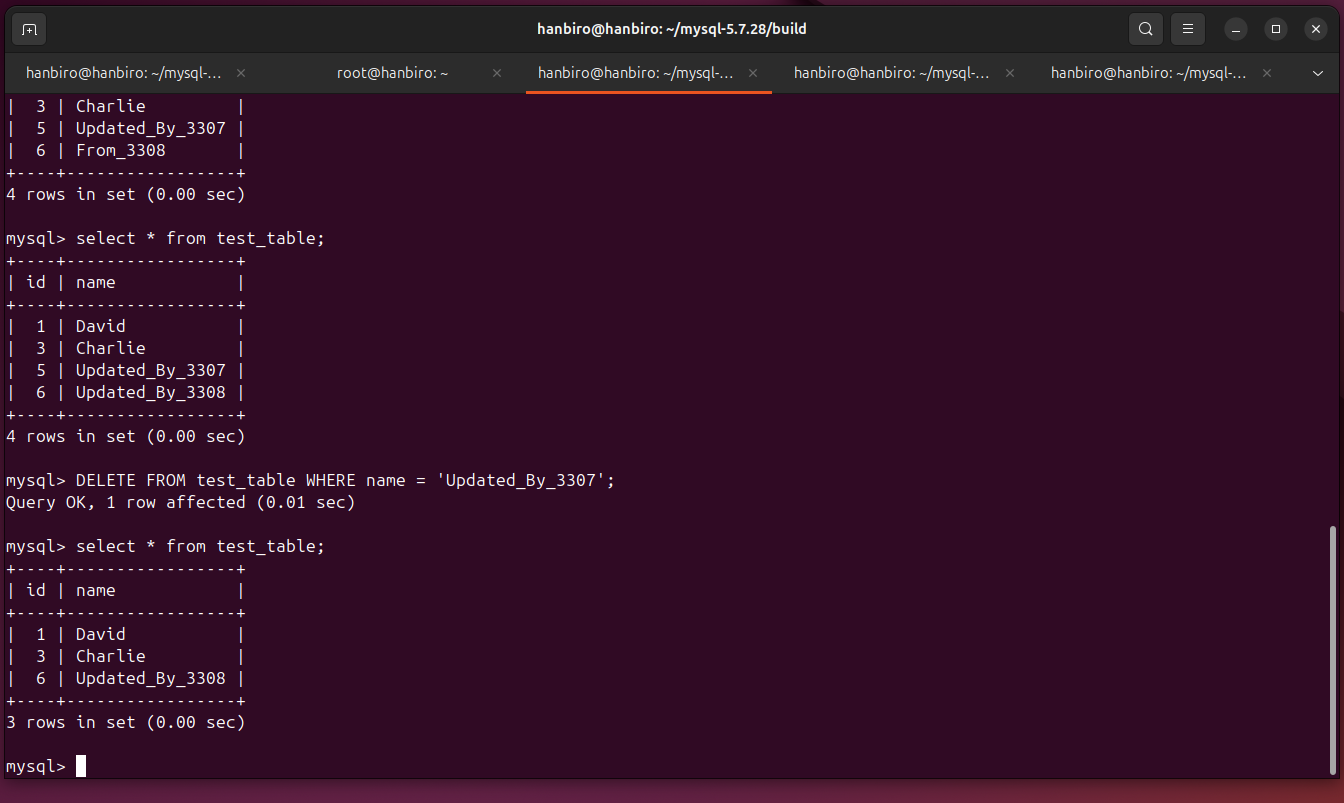
**2-way DELETE test**

**###** Test Delete from **Master -** check on **Slave:**

* on **Master**:

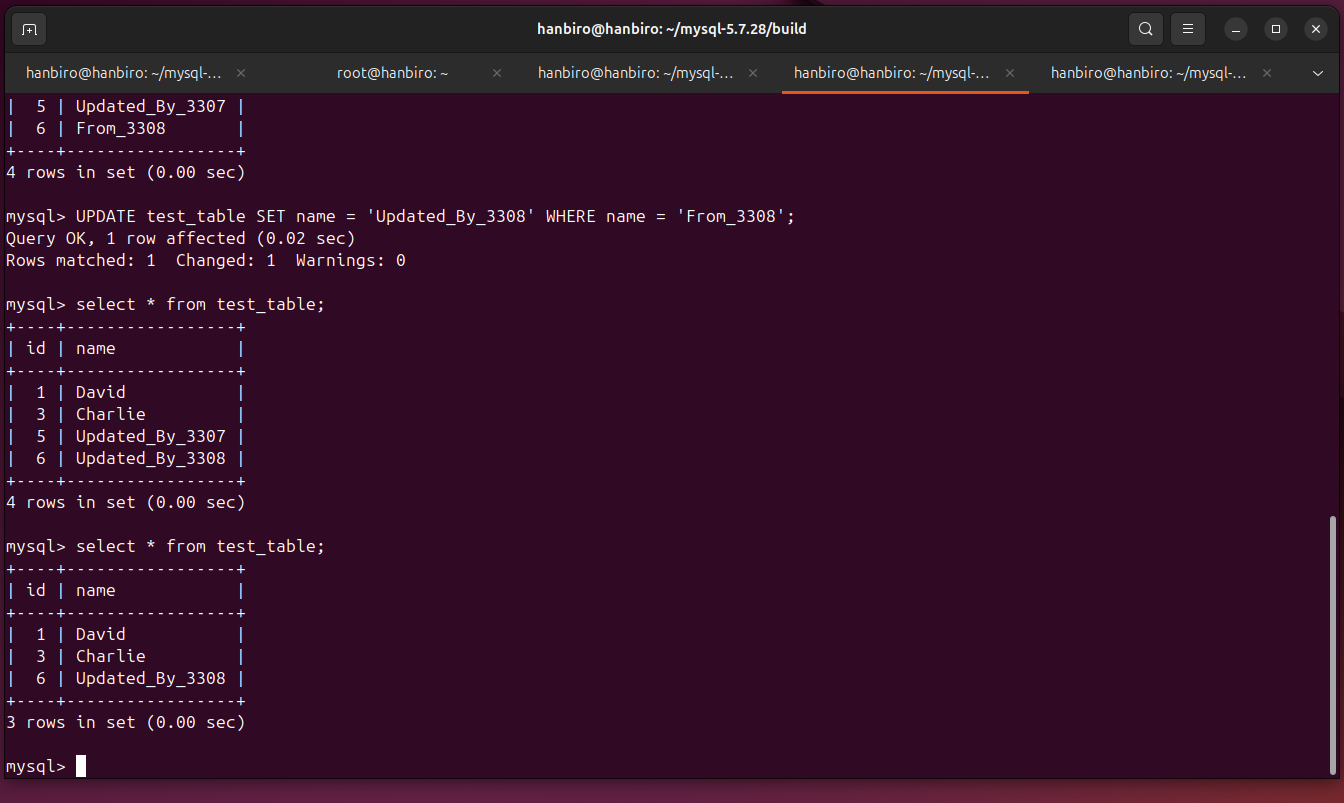
DELETE FROM test\_table WHERE name = 'Updated\_By\_3307';

select \* from test\_table;



* check on **Slave**:

select \* from test\_table;

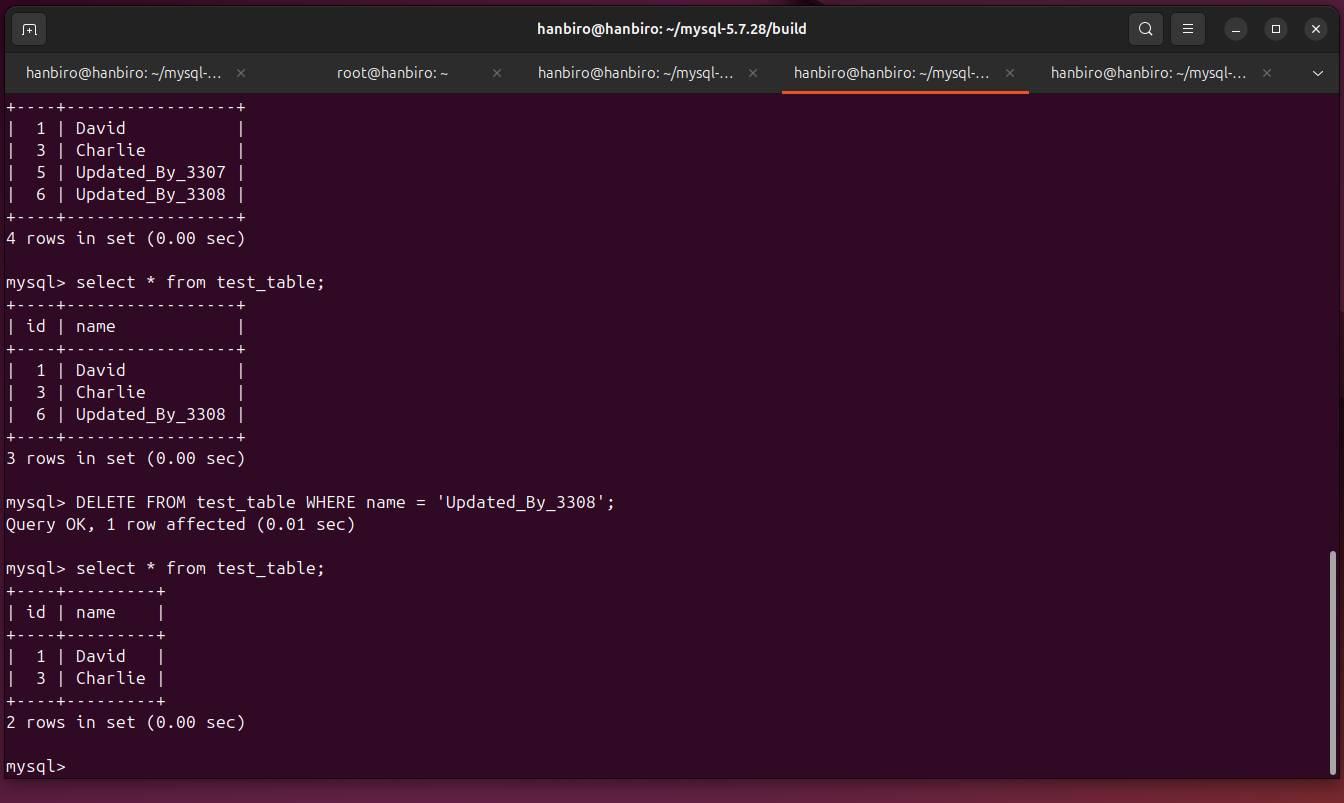


**###** Test Delete from **Slave -** check on **Master:**

* on **Slave**:

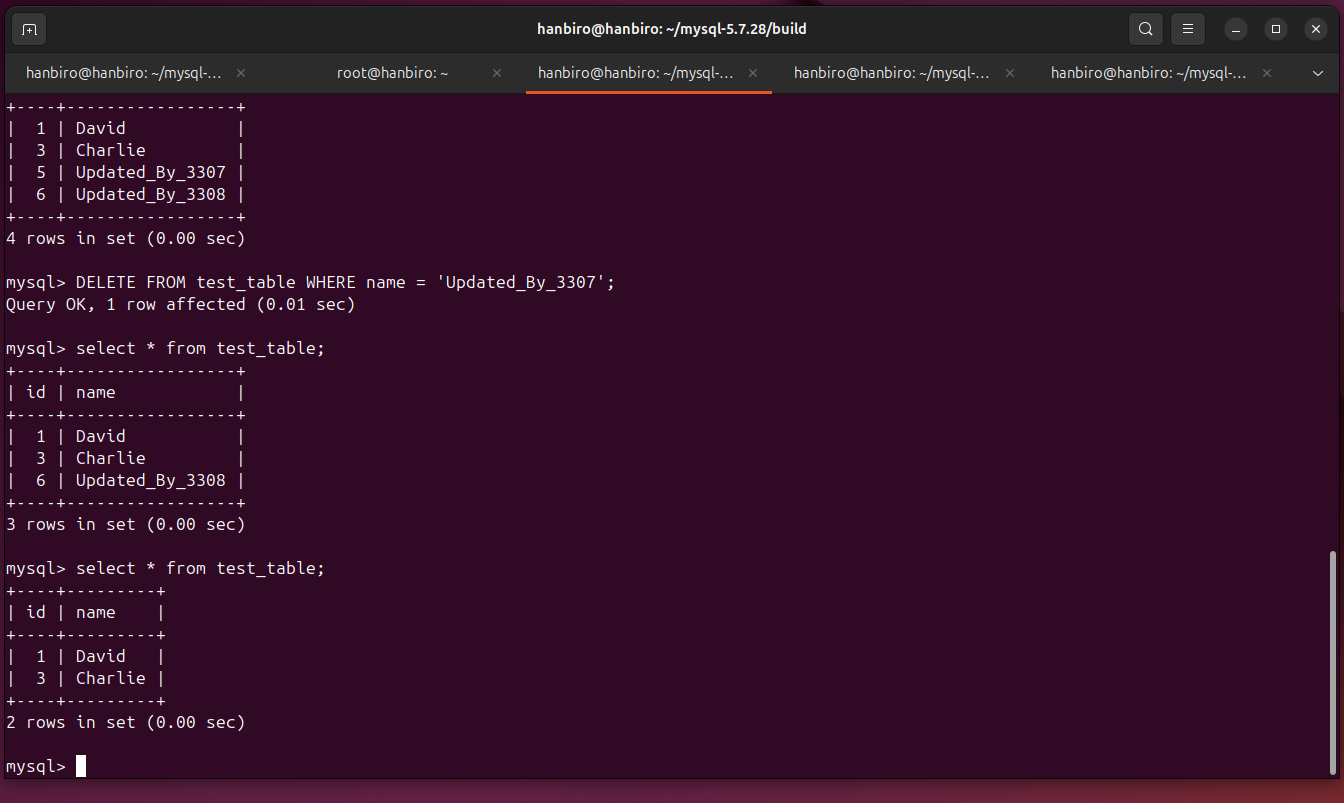
DELETE FROM test\_table WHERE name = 'Updated\_By\_3308';

select \* from test\_table;



* check on **Master**:

select \* from test\_table;



**Create replica user on master or slave?**

**Replica** users need to be **created** on **both Master and Slave** in the **Dual Master model**

* before switching to Dual Master (2-way replication), in the normal Master-Slave model, it is only necessary to create a replica user on the Master so that the Slave can connect and synchronize data
* when switching to Dual Master, because each node (Master) needs to connect and replicate to the remaining node, it is necessary to create a replica user on both Master and Slave to allow replication from this Master to Slave and vice versa. This helps set up two-way replication, ensuring that each node can synchronize data from each other without causing errors

Reasons for creating replica users on both nodes:

* the Master will create a replica user to allow the Slave to connect and synchronize data
* the Slave (when becoming the Master in the opposite direction) also needs a replica user so that the Master can connect and replicate from the Slave back

**Can we set to sync some specific data from master to slave or can we only set to sync all?**

In MySQL, it is possible to configure to only synchronize a portion of the data from the Master to the Slave — not necessarily to synchronize the entire database

This is done by configuring filter parameters on the Slave side, for ex:

* replicate-do-db: synchronize only specified databases
* replicate-do-table: synchronize only specific tables
* replicate-ignore-db: ignore some databases
* replicate-ignore-table: ignore some tables

Can add these lines to the Slave's /etc/mysql/**slave/my.cnf** configuration file, for ex:

[mysqld]

server-id=2

...

**replicate-do-db=replicated\_db**