Linux System Administration

PM-TRTNB319

Zsolt Schäffer, PTE-MIK

Legal note

These slides form an outline of the curriculum and contain multiple references to the official Red Hat training materials (codes RH124, RH134 and RH254), since students of this subject are also eligible for the named RH learning materials. Some diagrams and pictures originating from the RH courses will show up in this outline as there is an agreement in place which allows for our Faculty to teach this subject based on the Red Hat curriculum.

All other pictures and diagrams sourced from third parties are explicitly marked with a reference to the origin.

Legal note

All students of PTE-MIK who have officially taken the course „Linux System Administration" are eligible to download these slides from the Faculty's internal network, and are eligible for license keys for Red Hat System Administration online learning materials at rhlearn.gilmore.ca as part of their training program.

Do not share, redistribute, copy or otherwise offer these learning support materials, license keys or account information either for money or free of charge to anyone, as these materials contain **intellectual property.**

Unauthorized distribution is both against the law and university policy and will result in an in-campus inquiry, suing for damages and/or criminal prosecution by the University of Pécs, Red Hat Inc. and other parties.

MariaDB

Lecture 11

Chapter 1

History of MySQL

- Ancestors: TcX (Sweden) web development company created it's own database engine (1995) for internal use. It was based on mSQL and Unireg. Later released under the GPL license.
- TcX was later renamed to MySQL AB. Sun Microsystems bought MySQL. Oracle bought Sun Microsystems.
- Oracle already had a professional enterprise level database product: OracleDB. Development of MySQL somewhat died down.
- MySQL was already open-source and was available under the GPL license almost from the beginning. → Enthusiasts created a fork and continued development → MariaDB was born. MariaDB 5.5 is a compatible drop in replacement for MySQL 5.5.
- RH/Fedora still keeps both as a package in repositories.

Relational databases

- A relational database allows for storing data in persistent, structured way. It is organized as follows:
- A database consists of one or more tables, each describing an entity.
- Tables contain multiple records (lines).
- Each column stores a specific type of attribute for the record.
- Relations are usually created by linking entities with their IDs. For e.g. there is a table to store the attributes for all the products. Each product has an ID. In a table that stores invoices, invoiced items are not stored redundantly, instead they contain reference entries like: "twelve pieces of product 19". But a printed invoice will have lines with product name, color, price, etc... (Look up the JOIN sql operation).

Challenges

- Concurrent access (a simplified example): two transactions happen at the same time. Let's say a shared bank account with a balance of 100$ is being withdrawn from. The wife withdraws 20$ in one branch (process 1 reads balance, subtracts 20$ and calculates $80 as new balance, tries to write this balance back.) The husband withdraws 10$ at the same time in an other branch (process 2 reads the balance as 100$, subtracts 10$ and calculates a new balance of 90$, writes this back to DB).
- For one there will be a race condition, either 90$ or 80$ gets written last, and only the last write will stick, but we cannot know beforehand which one. Even if we put the race condition aside, the fact is: neither 90$ nor 80$ is a correct result for this operation. Solution: locking.
- Problem: What/how to lock? R/W lock? How is speed affected? (low speed → queue buildup → more concurrent access?)

Challenges

- Complex instructions and conditions, e.g.: A customer finalizes a lengthy order. All items have to be removed from stock, to prevent others from buying the same merchandise multiple times. Also at the same time customer has to be charged for the amount of purchase. Customer pays with a credit card, but for some reason money does not get through. How to cancel? Now both customer credit and stock inventory is inconsistent.
- Solution: Transactions. Problems: processing speed, memory need.
- Transactions enable for a complex set of instructions to either be fully carried out or the ability to return to the consistent state before the transaction started.
- This involves snapshoting and locking ranges of memory. (can be CPU/memory intensive, can affect caching, …)

ACID test

- Requirements for transactions:
  - Atomicity: Atomic, undivideable set of instructions. No partial execution allowed.
  - Consistency: The database will transition from one consistent state into another consistent state.
  - Isolation: The effects of a transaction are invisible to others until it is fully carried out.
  - Durability: Once a transaction finishes, it will be permanent, no partial rollback or leftover inconsistency can remain.

DB engines

- As you can see different application requirements can be accommodated with different level of complexity (CPU and memory needs), different set of features.
- MySQL resolves this by offering several database engines. You can select different DB engines per table.
  - XtraDB (InnoDB): supports transactions, per record locking, no space limit
  - Aria (MyISAM): smaller footprint, quite fast, no transactions, per table locking, easily portable (copy between systems)
  - Memory: very fast, volatile, does not write to disk
  - Archive: only supports SELECT and INSERT, optimal for storing log entries

DB engines

- Common storage engines (continued)
  - CSV: reads, stores and appends directly to "comma separated values" format text files.
  - MyRocks: Compressed storage, with reduced write amplification, optimal for flash drives, embedded systems (e.g. data loggers).
  - TokuDB: transactional engine for large workloads, that don’t fit in memory, with good compression.
  - Spider: provides sharding, partitioned to multiple servers.
  - Ndb: network clustering engine (not active in MariaDB)
  - Galera (not an engine): synchronous multi-master clustering, preferred cluster method for MariaDB.

Architecture, features

- MySQL/MariaDB serves the requests with a threading model, multi-core CPUs can be benefited from.
- Query results are cached, repeating SELECT queries are served from cache, if possible.
- MariaDB improved a lot and added some missing features recently: stored procedures, triggers, views.
- MariaDB still struggles with subqueries, and it is not based on object oriented concepts (ORDBMS)
- MySQL is the preferred DB back-end for the LAMP stack. For long it has been the fastest of open-source DBMS (now it is postgreSQL). Many web apps rely on it.
- For an enterprise quality open-source Object-Relational DBMS look up postgreSQL (it is also part of RH repos). Many business applications, like ERP systems rely on it.

Architecture, features

- It is common to run the DB server on the same host, that runs the web application code (for e.g. a web store written in php). This has the advantage of almost no latency in communication with the DBMS. It is recommended to use Unix local domain sockets instead of the localhost loop-back adapter in this case.
- In same cases, for performance or partitioning reasons, the DB server runs on a dedicated host. Cooperating applications send SQL queries and receive result through a TCP network connection with the DB server.
- Just as TCP listen ports can instantiate numerous sessions (virtual end-to-end pipes, each connects the server process with a different client), Unix domain sockets can do the same with almost no overhead. Of course the restriction is that communicating processes have to run on the same host (kernel) to use this feature.

Unix domain sockets

- From the programmers point of view, using local domain sockets is almost the same, as using TCP sockets.
- Unix local domain sockets are filesystem objects.
- From the system engineers point of view, using socket files requires a lot less from the system. A smaller latency and larger throughput can be achieved, because there is no TCP involvement: no packet ordering, no routing, no congestion/flow control, no retransmissions, no checksum calculation, no encapsulation happens.
- The loopback network adapter intentionally hides the fact that sender and receiver are on the same host, it is just a TCP/IP channel, like any other. Unix domain sockets allow for the sending thread to write the stream or datagrams directly into the receiving socket buffer.

MariaDB server and client

- The **mariadb-client** package group contains utilities for connecting to a MySQL/MariaDB server from the command line. It allows for the user to type and send SQL queries to the server, and displays the results as text tables.
- The client package also contains some libraries and connectors for userspace applications that rely on submitting MySQL queries.
- The **mariadb** package group contains the server part (that can store databases in files receive and respond to SQL queries through domain sockets or TCP sockets). It also contains a test database and provides some management/backup tools like the mysqldump command.
- A well known GUI client is MySQL Workbench.

my.cnf

- Both server and client store their configuration in **/etc/my.cnf**, but settings are separated to different sections.

/etc/my.cnf [excerpt]

```
[client]
port                    = 3306
socket                  = /var/run/mysqld/mysql.sock
[mysql]
prompt                  = '\u@\h [\d]> '
default_character_set   = utf8
[mysqld]
bind-address            = 0.0.0.0 #(single entry!)
port                    = 3306
socket                  = /var/run/mysqld/mysql.sock
datadir                 = /var/lib/mysql
tmpdir                  = /tmp
default_storage_engine  = InnoDB
character_set_server    = utf8
max_connections         = 505
max_user_connections    = 500
```

MariaDB installation

- Run **yum groupinstall mariadb** to install relevant packages. The package manager will create a technical user for the daemon (mysql, see /var/lib/mysql folder)
- Edit **/etc/my.cnf** with the text editor of your choosing.
- Run **systemctl start mariadb** and **systemctl enable mariadb**.
- Run **firewall-cmd --add-service=mysql --permanent; firewall-cmd --reload** to open port 3306/tcp. (default) Remember that mysql also has TCP Wrapper bindings.
- Run **mysql_secure_installation** to start the security setup wizard and set up a root password. Note that though they have similar names, the **mysql root** user is not to be confused with the **system root** user.
  Run **mysql -u root -p** to test your login.
- Complete practice **RH254 CH 9.2.** (MariaDB install)

MySQL client

- Use **mysql -u *username* -p -h *hostname*** to log in to a given server with a given user. The -p switch stand for password prompt. Otherwise mysql will not ask for a password, instead looks for it in environment variables ($MYSQL_PWD) or my.cnf. If -h switch is omitted, the local domain socket file will be used instead of TCP.
- Once logged in try the following commands to obtain general information:
  - **SHOW databases; USE *dbname*; SHOW tables;**
  - **DESCRIBE *tablename;***
  - **SHOW OPEN TABLES FROM *dbname;***
  - **SELECT *current_user*();**
  - **SHOW privileges; SHOW grants for *user@host;***

MySQL client

- Let's try some data definition commands:
  - **CREATE database first_db CHARACTER SET=utf-8;**
  - **USE first_db;**
  - **CREATE TABLE employees (name CHAR(20), job_descr CHAR(30), salary INTEGER);**
  - **ALTER TABLE employees ADD COLUMN entry_date DATE;**
  - **ALTER TABLE employees MODIFY salary FLOAT;**
  - **CREATE TABLE purchase_orders (quantity INTEGER UNSIGNED, date DATE) ENGINE= INNODB;**
  - **DROP TABLE purchase_orders;**

MySQL client

- Let's try some data manipulation commands:
  - **INSERT INTO employees (name, salary) VALUES ('John Smith', 250);**
  - **INSERT INTO employees (name, salary, entry_date, job_descr) VALUES ('Foo Bar', 350, STR_TO_DATE ('23-11-2017','%d-%m-Y'), 'junior engineer');**
  - **SELECT * FROM employees;**
  - **SELECT name, salary FROM employees WHERE salary > 300;**
  - **UPDATE employees SET salary=400 WHERE name='John Smith';**
  - **DELETE FROM employees WHERE salary > 380;**

MySQL users and privileges

- Data control commands (GRANT, REVOKE) will be discussed with MySQL users.
- Rules are stored as table entries, and are evaluated as ACL's, with the general rule being of the lowest priority, and the specific rule overriding the general rules.
- There are 5 tables involved, providing increased granularity of permission control in the following order: **user, db, host, tables_priv, colums_priv.**
- The user table basically defines the user accounts and corresponding passwords. The db table defines what user can do what operations on the databases. The host table refines control depending where the user logged in from. (Users can have different privileges depending on for e.g.: whether they logged in from the office LAN or from a public site.

MySQL users and privileges

- The tables_priv and columns_priv built-in tables allow for fine grained control over which user can have what rights on each table, or even each column. For e.g. a foreman can assign new work shifts to employees, but can not modify the salary column.
- You have two fundamental ways for managing users and access rights:
  - either use the CREATE USER, GRANT, REVOKE commands
  - or use basic data manipulation commands (like UPDATE, DELETE, INSERT) on mysql's built in user, db, host and other relevant tables. If choosing this solution you have to manually trigger the **FLUSH PRIVILEGES;** command to apply changes made to the permission settings.

MySQL users and privileges

- To create a user use one of the following schemes:
  - **USE mysql; INSERT INTO user (user, host, password) VALUES ('bob', 'localhost', password('secret')); FLUSH PRIVILEGES;**
  - **CREATE USER bob@localhost IDENTIFIED BY 'secret';**
- The latter command will create a user, set passwords, also set host related records and automatically flushes privilege cache.
- The newly created user can log in, but can not actually execute any commands.
- Host can contain the following example expressions: 'localhost', '192.168.20.40', '192.168.20.%', '%'

MySQL users and privileges

- To add permissions use the following scheme: GRANT *list_of_COMMANDS* ON *db_name.table_name* TO *user@host*; For e.g. **GRANT SELECT, INSERT ON first_db.employees TO bob@'localhost';**
- Practice: Look at the db table after previous command!
- You can use the * wildcard on *table_name* and/or *db_name.*
- As for the privilege list, you can use the ALL PRIVILEGES keyword. The USAGE keyword is a synonym for no privileges, it only allows for a user to log in, nothing else.
- Use the REVOKE command with the same logic to take privileges away from a user.
  E.g.: **REVOKE ALTER, DROP ON *.* FROM bob@'%'**
- Complete practice **RH254 CH 9.4.** (MariaDB users)

MySQL logs

- MySQL has it's own 4 separate logs. (unrelated to the system logger daemon)

/etc/my.cnf [excerpt]

```
[mysqld]
log_error                       = <hostname>_error.log
log_warnings                    = 2

slow_query_log_file             = <hostname>_slow.log
slow_query_log                  = 1
long_query_time                 = 0.5
min_examined_row_limit          = 100

general_log_file                = <hostname>_general.log
general_log                     = 0

server_id                       = 42
log_bin                         = <hostname>_binlog
binlog_cache_size               = 1M
max_binlog_size                 = 128M
expire_logs_days                = 5
```

MySQL logs

- MySQL has it's own 4 separate logs. (unrelated to the system logger daemon)
- Error log and general log is self explanatory.
- Slow query log holds the events when the processing of SQL queries took too long. This is useful for development purposes or to filter out users, SQL clients that use a substantial amount of system resources.
- The binary log contains all the data alterations. A DB can be rolled back to former state with special tools, when the binary log is available. But the main purpose of this type of log is to facilitate the replication of a server. (High availability services, backup servers.)
- The binary log can't be read directly, use **mysqlbinlog /var/log/mysql/mysql-bin.000001** to view contents.

Backup and restore

- When creating a backup of a DBMS you have to consider the role of memory caches. The state stored on disk files usually does not equal to a complete, consistent copy of all the table states.
- Therefore simply creating a copy of the directory that holds DBMS raw data (/var/lib/mysql) usually leaves you with an unusable backup.
- You can either make steps to ensure the consistent state of disk files and than create a **physical backup** of the backing-store. Or you can use the DBMS system to create dump of the current state to a separate backup file (called **logical backup**).
- Logical backups are clear text files containing SQL instructions, that if executed in order, will form a database matching the one that was backed up.

Backup and restore

- Logical backups are highly portable, and may even be compatible with other DBMS system, since they only contain standard SQL text instructions. Logical backups can be created while server is running, no interruption needed.
- Physical backups have a smaller footprint, and can be created faster that logical backups. Physical backups require the server to be brought offline or at least in a locked state to ensure consistency. Physical backups can also contain the logs and configuration files, while logical backups can't. Physical backups can only be ported to a narrow subset of versions of the same DBMS.
- Complete practice **RH254 CH 9.6.** (restore backup)
- Complete practice **RH254 CH 9.7.** (configure MariaDB)

Logical backups

- Use the **mysqldump -p -u root *db_name table1 table2* > /backups/backup_db_2017-11-27.sql** command to create a text sql dump of tables table1 and table2 inside db_name to a file called backup_db_2017-11-27.sql. The -u and -p switches work similar to the mysql command.
- If you want to create a backup of all tables, omit the table names. If you would like to backup everything, omit the db_name argument and add **--all-databases** switch. You also specify a list of databases when using the **--databases db1,db2**, … switch.
- Some additional useful switches: **--no-data** (only saves structure, but not the records), **--lock-all-tables** (prevent writing while dump happens), **--no-create-info** (only save records, do not create structure, assumes compatible existing structure), **--add-drop-table**, **--add-drop-database** (on restore, existing table or db will be dropped before restore operation)

Physical backups

- Create a copy of the mysql datadir (/var/lib/mysql by default) with cp, rsync, tar or similar (Refer to Lecture 12 for more) for. e.g.:
  cp -a /var/lib/mysql /backups/mysql/2017-11-27
- Or you can also choose to create an lvm snapshot of the volume that holds the data directory. Note that creating a snapshot takes the fraction of a second, later on you can use tar and compress or copy the directory from the snapshot, while DBMS resumed normal operation.
- To bring the disk files to a consistent state **always do this** before a physical backup: either use **systemctl stop mariadb** or use mysql client to flush and lock the tables the following way: **FLUSH TABLES WITH READ LOCK;** After snapshot: **UNLOCK TABLES;** or **systemctl start mariadb**. (See **RH254 CH 9.6.** for more.)

Reset mysql root password

- In case you forget the password of the **mysql root** user, you can perform the following emergency steps to reset the password.
- Stop mysql /mariadb (**systemctl stop mariadb**).
- Open up a dedicated terminal and execute: **mysqld_safe --skip-grant-tables** (this will keep running)
- **mysql -u root → use mysql; UPDATE user SET password=password('new') WHERE user='root'; FLUSH PRIVILEGES;**
- Now terminate the mysqld_safe process in the other terminal with **ctrl+c**.
- Start the daemon as usual: **systemctl start mariadb**
- In case a regular user's password gets lost, just use the root user to modify the user table with a new password.

Samba

Lecture 11

Chapter 2

Samba

- Microsoft Windows File and Printer Sharing services implement the Server Message Block (SMB) protocol. Common Internet Filesystem (CIFS) is a dialect of SMB, in practice these names are often used interchangeably.
- The name Samba (on some Linux systems smb or smbd) corresponds to the Linux implementation of the SMB protocol (and a set of relevant tools).
- Samba integrates perfectly with Windows networks. It can be both used as a client to services (shares) provided by a Microsoft product, both the other way around: a Windows computer can connect to a Samba provided share. In fact Samba can even work as Primary Domain Controller (PDC) for Windows workstations in an mixed-OS enterprise environment.

Samba security

- When creating a share, samba can follow one of the following 4 security schemes:
  - User level security: User authenticates to the server with a username/password upon session initiation. Once authenticated, all allowed shares can be accessed by the user.
  - Share level security: User authenticates against each shared directory separately.
  - Domain security: All authentication request are passed through the PDC. The samba server requires a machine account to be added to the Server Manager of the DC.
  - ADS security: Samba (vers. 4) can join as a native member to Microsoft Active Directory Services.

Defining SMB shares

- Install the **samba** package. Use your favorite editor on **/etc/smb.conf** file.

/etc/smb.conf [excerpt]

```
[global]
    server string = Samba Server version %v
    netbios name = server-example
    hosts  allow=172.25.0.0/24
    workgroup = Workgroup
    encrypt passwords = true
    guest account = nobody
    invalid users = root
    max protocol = SMB2
    passdb backend = tdbsam
    security = user

    vfs objects = acl_xattr
    map acl inherit = yes
    store dos attributes = yes

    allow insecure wide links = yes
```

Defining SMB shares

/etc/smb.conf [excerpt]

```
[homes]
    comment = %u's Home Directory
    browsable = no
    read only = no

[media]
    comment = Share for TV connected media player
    path = /shares/media
    valid users = bob,alice, @media
    read only = no
    guest ok = no
    create mask = 0660
    directory mask = 0770
    browseable = yes

[printers]
   path = /var/spool/samba
   print ok = yes
   printable = yes
   browseable =no
```

Enabling SMB shares

- After setting up the shares in smb.conf, make sure the shared directory has a proper SELinux context: **semanage fcontext -a -t samba_share_t '/shares(/.*)?'** **restorecon -Rv /shares**
- Sharing home directories through Samba requires the following SE Boolean set: **setsebool -P samba_enable_home_dirs on**
- Change the owner and mode of the shares directory, remember to set the SETGID bit for the directory: **chgrp -R media /shares/media; chmod -R 2775 /shares/media**
- Create corresponding Unix users. For e.g.: **useradd -s /sbin/nologin -G media john**
- And finally create a samba password (LM, NTLM hashes) for the user: **smbpasswd -a john**
- Enable the relevant firewall services: **firewall-cmd --add-service=samba --permanent; firewall-cmd --reload**

Enabling SMB shares

- This will open the following 4 ports: 137/udp (nmbd), 138/udp (nmbd), 139/tcp (nmbd), 445/tcp (smbd)
- The nmbd ports are required for NetBIOS name resolution to work. If you want to refer to Windows computer by their NetBIOS name, other than opening the ports you also have to start the nmb service: **systemctl start nmb**.
- Finally you may start the Samba service: **systemctl start smb**. You may also set the services for autostart: **systemctl enable nmb smb**
- Remember that changing smb.conf requires **systemctl restart smb** for changes to take effect. You can use **testparm** to check smb.conf and **pdbedit** to manage Samba users and passwords.
- Complete practice **RH254 CH 8.6.** (provide SMB services)

Samba client

- To identify a remote share: **smbclient -L *//server_name***
- This will list all the shares on the server that are set to browseable=yes. You can still mount the others, if you know the name of the share exactly.
- To perform a mount, consider the following example: **mount.cifs -o username=bob,password=secret //server_ name/share_name /mnt/samba**.
- Typing a password like this may not be secure, since it will be preserved in your Bash history. You can omit the password option, mount.cifs will ask you for a password from stdin, if required.
- You can also use the **-o guest** option, if the share allows anonymous access.
- Note the following options for local permission mapping: -o file_mode=0777,dir_mode=0777,uid=bob,gid=bob

Samba client credentials

- The **-o credentials=~/smb.cred** option to mount.cifs specifies a text file (e.g. smb.cred in your home directory) to read credentials from. This way you can authenticate without typing the password or leaving it in the shell's history. The format of a credentials file is the following:

~/smb.cred

```
username=bob
password=secret
domain=my_company
```

- The credentials file should only be readable by you: **chmod 600 ~/smb.cred**
- With a credentials file you can even do fstab mounts:

/etc/fstab [excerpt]

```
//serverX/media        /mnt/samba      cifs
          credentials=/secure/credentials.file 0 0
```

- Complete practice **RH134 CH 12.2.** (mount SMB shares)

Recommended reading

Lecture 11

Chapter 3

Relevant chapters in RH:

- RH134 CH12 (Samba)
- RH254 CH8 (File storage servers/Samba)

Read by next lecture:

- RH134 CH11 (NFS, autofs)
- RH254 CH8 (File storage servers/NFS)
- RH124 CH12 (Archiving and copying files)
- RH124 CH15 (Virtualized systems)

Use your rhlearn.gilmore.ca login to access the learning materials.

Thank you for your attention!

Lecture 11, PM-TRTNB319, Linux System Administration

Zsolt Schäffer, PTE-MIK