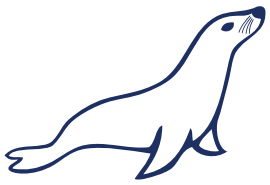
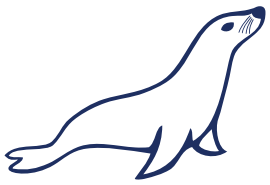


w: www.mariadb.com

e: info@mariadb.com

A Quick Start Guide to Backup Technologies



w: www.mariadb.com

e: info@mariadb.com

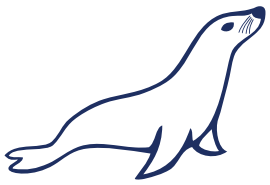
Objective Of This Guide

Data are as important for an organization as air. If, at any point, data and information become unavailable, the activities of modern organizations may even completely stop and lead to an unmanageable loss. A Database Administrator must think, not only to prevent extreme disasters (fire, flood…) but any possible hardware failure (system crash, hard disk failure) or user application failure (accidental DROP TABLE, or malformed DELETE FROM…) that will possibly lead to a data loss.
The most common procedure used to prevent data loss is the backup of a database. Backups are performed and scheduled in many different ways, according to the nature of service that the organization provides.

Backups can be defined as the second copy of the original database. They are produced by backup facilities and provided by DBMS or by operating systems. This second copy, or saved copy, is produced on separate storage media such as magnetic tape, external, remote or shared hard disk, or remote servers. Backups should be placed in separate and secure locations, where unauthorized persons do not have any access. The best way, to store a database backup would be online, where it can be accessed at any time and from any location.

There are several ways to perform backups. Each option must be evaluated carefully in order to achieve the safest and most reliable rescue in case of database failure, whatever reason is behind the failure.

This guide presents the most used backup and recovery techniques available for MariaDB and MySQL databases and helps the evaluation of the best backup solution, according to the organization hardware infrastructure, quality of service and resource availability.



w: www.mariadb.com

e: info@mariadb.com

Backup Types

Depending on the use of a database within the organization backups can be handled differently. They can be of a different type, or they can be scheduled in different ways.

MariaDB or MySQL databases are just files that are stored on a database server. For this reason, a backup and restore process can be pretty simple. No matter which application, tool or script is used, all of the backups will fit into two types: Logical or Physical.

Logical Backups

WHAT IT IS

A logical backup is executed by querying the MariaDB or MySQL server to obtain the database structure and the content information.

Usually, a logical backup is generated by a tool that writes all the SQL queries required to create the tables of that database, as well as all the insert queries required to place the information back into the database's tables.

PERFORMANCE

A logical backup is slower than a physical backup, because the server must access the database and convert the physical data into a logical format. If the output of a logical backup is written on the client side, the server must also send the logical format to the backup program, hence the overall performance is affected by this aspect.

LIMITS

The logical backup does not include logs or configuration files, or other database-related files that are not part of databases. On the other hand, backups stored in logical format are machine independent and highly portable.

IMPACT

Logical backups are performed with the MariaDB or MySQL server running.

ADVANTAGES

SQL dumps have the advantage of being easy to use and flexible, as the SQL statements can be manipulated as required using standard text tools, so it is easy to restore only certain records or tables. They are machine independent and highly portable.

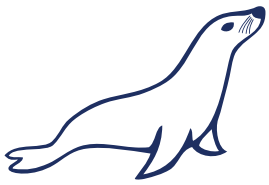
DISADVANTAGES

Backups in logical format are large, particularly when saved in text format, and often slow to create and restore.

Physical Backups

WHAT IT IS

The physical backup consists of exact copies of database directories and files. Typically, this is a copy of all or part of the MariaDB or MySQL data directory.



PERFORMANCE
A MariaDB or MySQL Raw Backup is performed quicker than a logical backup, because it does not translate the contents of the database into human readable SQL queries. It simply copies data files from one storage to another and the output is more compact than a logical backup. In addition to databases, the backup can include any related files such as log or configuration files.

LIMITS
Data from MEMORY tables cannot be backed up in physical format because their contents are not stored on disk.

IMPACT
Physical backups can be performed while the MariaDB or MySQL server is not running. If the server is running, it is necessary to perform appropriate locking so that the server does not change the database contents during the backup.

ADVANTAGES
A physical backup can be performed using utilities and tools, but also manually by copying data files to a backup storage device. Raw backups are usually faster to restore: the restore procedure is basically a simple copy of the binary files.

DISADVANTAGES
Physical backups are portable only to other machines that have identical or similar hardware characteristics.

Snapshot Backups

Snapshots are a third option, a “special” way to perform hot physical backups that rely on the functionality of the operating system or the storage solutions.

Some file system implementations enable “snapshots” to be taken. These provide logical copies of the file system at a given point in time, without requiring a physical copy of the entire file system. (For example, the implementation may use copy-on-write techniques so that only parts of the file system modified after the snapshot time need to be copied).

MariaDB or MySQL itself does not provide the capability for taking file system snapshots. This capability is available through third-party or OS solutions such as Veritas, LVM, or ZFS.

Logical (Data dump) vs. Physical (Raw) backups

A data dump results in a sequence of SQL statements, which can be executed against any database to recreate the database structure and the data itself. However, data dumps incur in a lot of overhead with extra SQL syntax, and result in larger data files. They are also more CPU intensive, and most importantly, they require a full index rebuild when the data is being restored. Restoring data from the logical backups could take a long time and it could be difficult to predict the size of logical format. Since they are usually text files, logical backups can be highly compressed.

Arguably, the most efficient way to backup a database is through a raw snapshot of the MariaDB and MySQL files as they exist on disk. Because there are no conversion steps, the process is more efficient

than a data dump. For example, in order to perform a backup of a MyISAM table, the data and the index files must be copied; for InnoDB, the entire tablespace and the associated transaction logs must be copied. Physical backups and especially restores are much faster - usually 5 times faster than the logical backup, and 20 times faster than the logical restore. Furthermore, the exact size of the physical backup is always known, since it is an exact copy of a database.

Raw backups can be recovered ONLY to the same version of the MariaDB or MySQL server on the same operating system as the original data. It means that the chances to recover raw backup images of MariaDB or MySQL to another managed hosting provider are not very high and DBAs should take that into consideration when choosing raw vs. logical backup

Backup Methods

Backups can have an impact on the activity of a live server and sometimes a backup or a restore activity can cause services disruption.

Backups can be then defined in three main groups depending on the impact they have on the database server activity and on how it blocks the database operations.

Cold Backup (Offline)

A cold backup is executed when database is offline, not accessible for updating and the users cannot do anything on the database: the operations are stopped. This procedure is also called offline backup. Normally, a cold backup is taken at the end of a working day or at weekends, when for some operations the database can go offline. A cold backup is suitable in environments where the database does not run for 24-hours a day. It is not considered as a *blocking backup* as it is performed while the system - meaning the entity that provides the service, not the machine or the operating system - is down. Offline backup procedures are simple because there is no possibility of interference from any client activity.

Warm Backup (Oline-Offline)

In a warm backup, the server is running but the data cannot be modified while you the database files are accessed externally. The system and the operations are limited, typically to read-only mode. It is a *blocking* backup because it limits and partially stops normal operations.

Hot Backup (Online)

A hot backup is taken while the MariaDB or MySQL instance is running and the applications are reading and writing to it. It does not require any downtime or the service to stop. A hot backup is required in environments where the database needs to remain online 24-hours a day and 7-days a week. A hot backup can be also identified as “dynamic” or “active”. Hot backups do not interfere with the transactions that are executed while it is running and for this reason it is also called *non-blocking backup*.

A hot backup is more complicated than a simple copy of data files: it must include any data that was inserted or updated while the backup was in process; it must exclude any data that was deleted while the backup was in process; and it must ignore any changes started by transactions but not committed.

To lower the impact of a hot backup on a server, this procedure should only be executed during low working hours.

A similar distinction between online and offline procedures applies for recovery operations, that share similar characteristics. However, it is more likely that clients will be affected by an online recovery than by an online backup, because a recovery requires stronger locking. During the backup operations, clients might be able to read data at the same time they are backed up. A recovery operation modifies the data and does not just read it, so clients must be prevented from accessing data while it is being restored.

Online vs. Offline backups

Online backups are often the preferred method as the database downtime may not be an acceptable option. Having said that, offline backups are usually faster and less error prone, as DBAs and users should not worry about running transactions, table locks, orphaned processes, and other consistency problems; care must be taken to impose appropriate locking so that data modifications that would compromise backup integrity do not take place. If it is affordable to have a brief period of downtime and clients are not adversely affected because the server is unavailable during a backup, or if a master-slave replication is available, offline backup is the way to go.

Not all the MariaDB or MySQL storage engines fully support online backups. InnoDB/XtraDB and NDB fully guarantee data consistency and fully support online backups; in case of MyISAM, the DBAs must set the database in read-only mode.

Local and Remote Backups

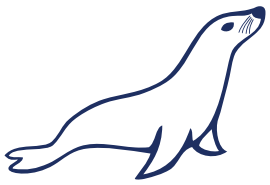
A *local backup* is performed on the same host where the MariaDB or MySQL server runs, whereas a *remote backup* is executed from a different host. For some types of backups, the backup can be initiated from a remote host even if the output is written locally on the server.

The question crossing most DBAs' mind is: should the data be stored locally or remotely? To be fair, this is the crucial point of all data backup plans. There are pros and cons to each method.

Local Data Backup

Most companies rely on DVD, external hard disk or tape drive data backup solutions locally, i.e. they execute a backup with this storage available on the same database server. These solutions provide a fast, cost effective storage for many small companies and are also very easy to operate. In the case of any data being lost, it is simply a matter of restoring the data from a previous backup point and the user can then continue working. Local backups are, also for this reason, the default location for all the tools.

In case of local backup, it is important to choose which is the most convenient media to use. Hard drives are faster, more reliable and easier to use than tapes or optical disks, but tapes and optical disks



have been the favorite choice for many years, due to their cost and to the fact that large hard drives were rare and very expensive; now the cost per TB is in favor of SATA HDDs.

The data backup, however, is only as safe as the building it is in. What happens in the case of fire, burglary, flooding or storm damage or just a disk failure? So, if local backups may be faster, it is necessary to perform extra steps to secure backup data files.

Remote Data Backup

Remote backups can be performed on remote servers within the same infrastructure or on remote file backup services that allow the logging into a remote drive and via a web-based interface to upload all backups and critical files.

Knowing that the data is being stored in a secure facility that is deliberately protected against harm, allows much greater peace of mind.

The weak point here however is the data transfer speed. Unless they operate on a high speed LAN backbone, the remote backup can be a complete waste of time as it is tied to the maximum upstream speed of the available network - which can be surprisingly slow, depending on the IT infrastructure. Furthermore, firewalls blocking uploads on certain ports can quickly make a backup a real headache and slow the operations down to an unreasonable point.

Backup Format

Whatever is the backup type and method used, a further step is to decide whether the backup should be compressed or it should be left as it is.

Plain Backup

A *plain backup* is a backup where no form of compression is applied to the final output. Plain backups can be quick to execute, but they may require a large amount of storage that sometimes make the result of the backup operations difficult to handle.

Compressed Backup

A *compressed backup* is a form of backup where a compression algorithm is applied to the final output. In some cases it is also possible to apply encryption algorithms to the output data.
Compressing backups can make a good sense in terms of backup and recovery speed as well as space needed, but they can also be a serious bottleneck, depending on the circumstances and the approach used. This is mainly true for medium and large size databases; for databases below 100GB in physical size, the performance of the compression algorithm is usually not a big issue.
One important point is to decide where to apply the compression: on the source (local backup) or on the target (remote backup) and which compression software to use.

Compression at source is the most typical approach and it works well, though it takes extra CPU resources on the source server in additional I/O resources which may not be available, especially for CPU bound MariaDB or MySQL database servers. The benefit in this case is less space requirement; the

backup is stored locally, as well as less network bandwidth requirements in case the backup is executed on network storage.

Compression at destination offloads the source server (though it may run out of CPU resources itself, if it is the target for multiple backups), but the network bandwidth requirements to transfer uncompressed backups are higher.

The classical tool used for backup compression is the *gzip* command or some built-in *zip libraries* – they exist almost everywhere, they is stable and relatively fast.

In many cases however, it might not be fast enough and become the bottleneck for the whole the backup process.

Backup Objects

Complete Backup

A *complete backup* includes all the data managed by a MariaDB or MySQL server at a given point in time. All the database schemas, tables and partitions are affected by the backup operations.

Partial Backup

A *partial backup* is applied to a subset of objects of a database. DBAs may decide to backup only certain schemas, tables or partitions.

Backup Strategy

Full Backup

A *full backup* is a consistent snapshot of a database at a given time. DBAs perform a full backup when they we to to create a backup that can be restored on its own on a recovery server.

Incremental Backup

An *incremental backup* consists of the changes made to the data during a given time span (from one point in time to another). DBAs perform incremental backups when the dataset is to large to perform a full backup every time. In order to be restored on a recovery server, DBAs need to collect the latest full backup and all the incremental backups, up to a given point in time.

MariaDB and MySQL have different ways to perform full backups, such as those described in the next section. Incremental backups are made possible by enabling the server's binary log, which the server uses to record data changes. According to the MySQL documentation, by enabling binary logs the server will be affected by a performance hit of less than 1% (http://dev.mysql.com/doc/refman/5.5/en/binary-log.html), but this value highly depends on the type of operations executed on the serve. For example, databases with heavy write operations will be highly affected by the use of the binary log, whilst database performing primarily read operation will experience a lower impact when the binary log is enabled.

To better clarify the relationship between backup objects and Strategy, each backup object, no matter if it is complete or partial, can be either full or incremental.

Restore

Database Clone

A *database clone* is a complete and separate copy of a database system that includes the business data, the DBMS software and any other application tier that makes up the environment. Cloning is a different kind of operation to replication and backups, since a cloned environment is both fully functional and separated by original environment. Additionally, the cloned environment may be modified at its inception, due to configuration changes or data subsetting.

The cloning refers to the *replication* of the server in order to have a backup, to upgrade the environment.

Full Versus Point-in-Time (Incremental) Recovery

A *full recovery* restores all data from a full backup. This operation restores the server instance to the state prior to the backup.

An *Incremental recovery* is a recovery of changes made during a given time span. This is also called *point-in-time recovery*, because it makes a server's state current up to a given time. A point-in-time recovery is based on the binary log and typically follows a full recovery from the backup files that restores the server to its state when the backup was made. Then the data changes written in the binary log files are applied as incremental recovery to redo data modifications and bring the server up to the desired point in time.

Backup Options

When the target scenario is clear, it is easy to go through all the available options and solutions to have a safe backup and restore of the MariaDB and MySQL data. Some options are MariaDB and MySQL specific (they are utilities that are provided with MariaDB and MySQL), others come from the Open Source community, others are commercial tools. A special section has been dedicated to backup manager tools: these tools do not perform a backup directly, but they can schedule, organize, archive and check the results of a backup operation.

mysqldump

mysqldump is a command line utility provided with MariaDB and MySQL that automates the process of taking a SQL dump. A database dump contains the table structure and data in SQL text (ASCII) format. It is a *logical*, and depending on what storage engines are used it can either be a *hot backup* for InnoDB only or a *warm backup* for MyISAM. *mysqldump* can also be used to generate files in CSV format, delimited text, or XML format.

How Does a Dump Work?

For each database schema and table a dump performs these steps:

1. LOCK TABLE *table name*

2. SHOW CREATE TABLE *table name*
3. SELECT * INTO OUTFILE *temporary file*
4. Write the contents of the temporary file to the end of the dump file
5. UNLOCK TABLES

Depending on what storage engines are used, *mysqldump* can have a behavior that slightly differs from the standard dump backup execution: this means that for the InnoDB storage engine, it is not necessary to lock the database tables, whilst it is mandatory for MyISAM tables. This is not automatic though: a DBA must be aware of the nature of the database and of the used engines and execute *mysqldump* with parameters that take this into consideration.

Example:

This is the mysqldump command used to backup all engines:

```
$ mysqldump --lock-all-tables [--flush-logs [--master-data]]
           [--all-databases | db_name] > bck_file
```

This is the mysqldump command used to backup InnoDB tables

```
$ mysqldump --single-transaction [--flush-logs [--master-data]]
           [--all-databases | db_name] > bck_file
```

Advantages

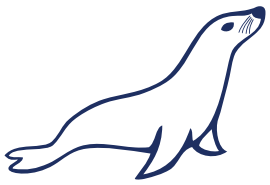
1. Platform independent
2. Portable for all MariaDB and MySQL® tables
3. Backup everything or just certain database tables
4. Backup only the DDL
5. Optimize the dump for a faster restore
6. *mysqldump* can indirectly detect corrupted data files: when a MySQLDump is taken, it is known that the data must be in a good state or an error would be generated during the dump process.
7. Corrupted databases can be corrected by looking at the contents of the dump.
8. *mysqldump* can connect to a local or remote servers: SQL output files, dumps can be local or remote, or the output can be generated on the client.

Disadvantages

1. It can be slow for large DBs and generate larger output files.
2. A full backup is executed every time. There is no way to do an incremental backup with a SQL Dump. This means that a backup can be very time consuming, especially on larger databases.
3. When using MyISAM, a dump holds a global read lock on all tables, blocking writes from other connections for the duration of the full backup. The locking can be optional. If the locking is not performed, there is no consistency in the backup.
4. Restoring only desired databases or tables requires the editing of the SQL Dump file. Editing raw database dumps is very time consuming and error prone.
5. Depending on several factors (for example the number of indexes), the restore time may be significantly longer than the backup time.

Conclusions

If the data set is relatively small (for example, less than 10GB), then *mysqldump* will probably work great. It is easy, it is online, it is very flexible and with an accurate use of options it is possible to avoid the risk of non-consistent backups.



It is not uncommon for database administrators to take a periodic database dump (e.g. weekly) even when they might be using a more advanced backup method for more frequent backups. Because of the readable SQL text nature of the database dump, *mysqldump* provides a peace of mind to the administrator.

mysqlhotcopy

mysqlhotcopy is a command line utility provided with MariaDB and MySQL that automates the process of doing a lock and copy on MyISAM tables that applies only to MyISAM and ARCHIVE tables. The utility performs a physical backup and despite its name it can be only warm (i.e. during the backup operations the tables are in read-only mode).

How Does MySQL Hot Copy Work?

6. FLUSH TABLES WITH READ LOCK
7. Full copy of MyISAM Tables (e.g. cp or scp)
8. UNLOCK TABLES

Example:

This is the mysqlhotcopy command used to backup MyISAM tables to a destination directory

```
$ mysqlhotcopy [--flushlog] db_name ... bck_directory
```

Advantages

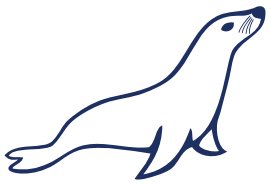
1. Fast Backup (faster than a SQL dump).
2. Fast restore (just a copy of MyISAM files).
3. It generates an exact copy of the original tables
4. It can backup all the database or single tables

Disadvantages

1. It works only for MyISAM and ARCHIVE tables
2. It can be performed only on the same machine where the database directories are located.
3. It only runs on Unix and Linux servers.
4. It holds a global read lock on all tables blocking writes from other connections for the duration of the full backup.
5. The global lock is held as long as it takes to copy all of the MyISAM files. For mission critical databases this means failed queries while the database is unavailable and queries time out waiting on the lock.
6. The restore is not straightforward. The files must be carefully copied back into the MariaDB or MySQL data directory and the permissions must be changed. This can be dangerous and mistakes can occur. A less dangerous operation is to shutdown the MariaDB or MySQL daemon and copy MyISAM files back into the data directory.
7. It performs only local backups. It connects to the server to lock it against data modifications and then copies local table files.

Conclusions

The challenge with *mysqlhotcopy* is that it must hold table locks for the duration of the entire MyISAM backup. This means that the database is unavailable while the tables are being copied. For most production environments these limitations make *mysqlhotcopy* useless, unless it is used during maintenance periods where down time is acceptable. mysqlhotcopy is sometimes used with great success to take a backup copy of a MySQL Slave replica in larger production environments.



Binary Logs

MariaDB and MySQL Binary Logs must be enabled and contain a record of each SQL query executed against the database that changes data since the logs where last flushed (new log file started). The log contains queries like *UPDATE* and *DELETE*. The log does *not* record *SELECT* or *SHOW* queries for example, because they do not change the database.

Binary logs can be used to perform a logical, incremental, hot and not locking backup.
The binary log files provide the information needed to replicate changes to the database that are made subsequent to the point at which a backup was performed. It can be used for point-in-time recovery. It can have an impact on operations as all the changes in the DB are logged in real time.

How Do MariaDB and MySQL Binary Log Backups Work?

With the binary logging enabled, the MariaDB and MySQL binary log files might look like:

```
localhost-bin.000001 localhost-bin.000002 localhost-bin.000003 localhost-bin.index
```

For each database and table, the backup program executes:

1. LOCK TABLES
2. FLUSH LOGS
3. SHOW CREATE TABLE *table name*
4. SELECT * INTO OUTFILE *temporary file*
5. Write the contents of the temporary file to the end of the dump file
6. UNLOCK TABLES

How the FLUSH LOGS Query Works?

When the *FLUSH LOGS* query is executed, MariaDB and MySQL starts a new binary log file to record queries in. So now the binary log files look like:

```
localhost-bin.000001 localhost-bin.000002 localhost-bin.000003 localhost-bin.000004
localhost-bin.index
```

Incremental queries made after the backup operation appear in the localhost-bin.000004 binary log file.

Restoring with Binary Logs

The procedure to restore from a point in time using binary logs is:

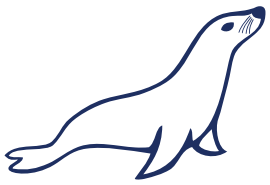
1. Restore the database from the last SQL Dump completed before the desired recovery point.
2. Use the *mysqlbinlog* utility to restore to the desired point in time. The *mysqlbinlog* utility converts the events in the binary log files from binary format to text so that they can be executed or viewed.

Advantages

1. It allows point-in-time restore right down to individual queries.

Disadvantages

1. It requires a full backup to be executed periodically. This means that a backup can be very time consuming, especially on larger databases.
2. Restoring only desired databases or tables requires the editing of the SQL Dump file.
3. Restoring to point in time with the *mysqlbinlog* utility can be complicated.



Conclusions

The use of binary logs for backup purpose can be a little complicated and goes through several manual steps; on the other hand, binary logs can be really powerful. For example, it is possible to pipe the result from the *mysqlbinlog* utility to a file, remove bad queries and restore it. It is also possible to limit the list of returned queries with start and end time and to restore the database to the state as it was on a specific date and time.

Replication

MariaDB and MySQL replication allows physical or logical backup, it is non-blocking and portable. Replication can be paused on slave to allow Point-in-time backups. Long backup operations can proceed without disrupting other servers. Backup disk I/O is away from customer facing servers and has a low impact on normal server activity. A challenge some deployments experience is that for larger databases or especially for those that are write intensive, long pauses in replication required for backup can be problematic.

Advantages

1. Backups can be both physical and logical
2. Along with *mysqldump*, it can be offline and non blocking
3. Backups are portable
4. The environment is flexible and easy to work with

Disadvantages

1. Not easy to manage for large data
2. On write intensive applications, replication can take some time to recover after a pause.

Conclusion

For larger systems is often the preferred solution along with *mysqldump* utility.

Physical Backup

An alternative backup solution is to perform a physical backup (also called *binary backup*), which reads the database files using the file system interface rather than the SQL statements. Typically, this is a copy of all or part of the MariaDB and MySQL data directory. Physical backup tools include file system-level commands (such as **cp**, **scp**, **tar** and **rsync**),

Advantages

1. Physical backup methods are faster than logical because they involve only a file copy without conversion.
2. Output is more compact than for logical backups
3. Backup and restore granularity ranges from the level of the entire data directory down to the level of individual files. This may or may not provide for table-level granularity, depending on storage engine. (Each MyISAM table corresponds uniquely to a set of files, but an InnoDB table shares file storage with other InnoDB tables.)

Disadvantages

1. Backups are portable only to other machines that have identical or similar hardware characteristics
2. Backups can be performed while the MariaDB or MySQL server is not running. If the server is running, it is necessary to perform appropriate locking so that the server does not change database contents during the backup.

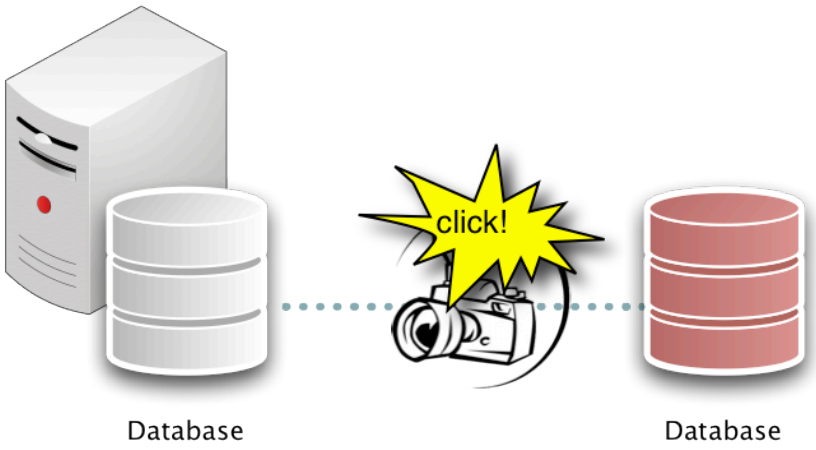
Conclusions

Note that the files may not be transactionally consistent at the time of the copy. Making them consistent is deferred until a database is restored from the backup. Making it consistent is similar to what happens when a database server restarts from an abrupt shutdown, like when a power failure occurs while the server is busy.

It is a physical copy of data files or data directory. It is a cold backup on the main system.

Storage Specific Backup-Snapshots

These are backup procedures that are usually associated to snapshots. They can be considered as hot backups as they lock the data in read only mode just for a fraction of a second.
Being storage specific, backup snapshots are often managed by third party solutions, or in some circumstances they came as specific operating system services.



Advantages

1. It is almost a hot backup: **i**n many cases it is possible to perform this type of backup while the application is running. No need to shut down server, make it read only or anything like it.
2. It works with MyISAM and InnoDB.
3. It provides a fast backup: it simply executes a file copy in the binary form.
4. It has low overhead: Being a file copy, the overhead to the server is minimal.
5. It is Easy to integrate: easy to compress, backup to tape, FTP or any network backup software – it is easy as just a file copy is needed.
6. It provides fast recovery: the recovery time is as fast as putting data back and standard MariaDB or MySQL crash recovery, and it can be reduced even further.

Disadvantages

1. It needs to have snapshot compatibility with the recovery server.
2. It may need root access to the system. In some organizations, DBA and System Administrator are different individuals and from different departments, which might not like to trade access rights between each other.
3. The downtime is hard to predict. This solution is often mentioned as a *hot backup*, but it is hard to estimate when it is hot and when it is not – *FLUSH TABLES WITH READ LOCK* may take quite a while to complete on systems with long queries.
4. It may cause problems with data on multiple volumes. If the logs are split on separate devices or if the database is simply spanned across multiple volumes, it might be hard to get a consistent snapshot across all the database. Some systems though may be able to do atomic snapshot of many volumes.
5. Secondly, although the captured snapshot may be in a known state as far as the file system is concerned, it may not contain information that was in memory at that time.

6. It may require a long crash recovery operation at restore time instead of at backup time.

Conclusions

Backup snapshots can be performed in many different ways, but the overall goal is always to provide you a fresh volume, which consistently matches the state of the volume at the time the storage is created. What is really needed is the ability to create atomic snapshots of the volume, which can be later mounted in the same way as on the original file system.

Snapshot cannot be considered as a full backup. It is good to restore the whole lot pretty fast, but they would not protect from data corruption (which can occur days, weeks or months before the backup has been performed). Furthermore, it is not possible to restore just one table, or only a few rows. However, it is possible to restore the backup on another system (letting InnoDB do any necessary crash recovery) and then check tables for corruption.

A common approach is to perform logical full backups on a regular basis and a rolling buffer of snapshots at a regular interval in order to be able to fast recover data in case of disaster.

Snapshot Technologies

There are several technologies to look at in order to perform snapshot backups. Here is a list of the most relevant or popular.

LVM – Linux Logical Volume Manager

LVM is a Volume Management system that has **built-in snapshot functionality**. A big disadvantage of LVM is that the MariaDB or MySQL storage must be configured on a LVM Volume and it must have a pre-configured dedicated storage space for snapshots. This means that the conversion of an existing MariaDB or MySQL installation on Linux to use LVM can be a large effort. Also, some older versions of LVM do not support freezing the file system in a consistent state. This means that they should not be trusted for online MariaDB or MySQL backups. Sometimes it is necessary to patch LVM with the VFS-lock patch to get consistent file system snapshots.

MyLVMbackup

MyLVMBackup is a tool for quickly creating backups of a MariaDB or MySQL server's data files. To perform a backup, *MyLVMBackup* obtains a read lock on all tables and flushes all server caches to disk, creates a snapshot of the volume containing the MariaDB or MySQL data directory, and unlocks the tables again. The snapshot process takes only a small amount of time. When it is done, the server can continue normal operations, while the actual file backup proceeds.

The LVM snapshot is mounted to a temporary directory and all data is backed up using the *tar* program. By default, the archive file is created using a name of the form *backup-YYYYMMDD_hhmmss_mysql.tar.gz*, where *YYYY*, *MM*, *DD*, *hh*, *mm*, and *ss* represent the year, month, day, hour, minute, and second of the time at which the backup occurred. The default prefix backup, date format and file suffix can be modified. The use of timestamped archive names allows you to run *MyLVMBackup* many times without the danger of overwriting old archives.

Alternatively, instead of *tar*, *rsync* can be used. The process is nearly identical, with the exception that the file suffix is not used. *tar* backup is primarily developed for performing local backups. The *rsync* backup can perform both local backups as well as backing up to a remote server using *rsyncd* or *rsync* via *SSH*.

MyLVMBackup also supports creating backups by using *rsnap*, which is a wrapper around *rsync* to automatically maintain and rotate a given number of last backups (7 by default). It utilizes hard links to link to unchanged files for saving disk space.

Requirements

It is required to run MyLVMBackup on the same host where the MariaDB or MySQL server runs. Even though MyLVMBackup communicates with the server through a normal client connection to obtain the read lock and flush data, it performs the actual backup by accessing the file system directly. It is also a requirement that the MariaDB and MySQL server data directory resides on an LVM volume. (It is, however, a good idea to do the LVM backup to a different partition than the one where the data directory resides. Otherwise, there is a good chance that LVM will run out of undo space for LVM snapshot maintenance and the backup will fail.)

It requires also other software tools installed such as *Perl 5*, *GNU tar*, *gzip* and *LVM* utilities. Optionally, *rsync* or *rsnap* may be required instead of *tar* and *gzip*, depending on which backup type is required.

Linux Hot Copy (R1Soft)

R1Soft Hot Copy (*hcp*) is a tool for taking point-in-time snapshots of any Linux block device that does not require *LVM*. *Hot Copy* is typically much easier to install on an existing MariaDB and MySQL deployment when compared to *LVM* and it requires no dedicated snapshot storage. The *R1Soft Hot Copy* utility creates an instant point-in-time volume snapshot of any block device while the system is running without interrupting applications. It works on almost any Linux block device. It is like *Volume Shadow Copy* for Linux.

It uses existing free space on disk to maintain snapshots and it has better performance compared to *LVM* snapshots. It keeps multiple snapshots of each disk or volume.

Veritas VxFS (Symantec)

Veritas is an extent-based file system. *VxFS* is used as the primary filesystem of the HP-UX operating system, where it is called *JFS*. It is also supported on AIX, Linux, Solaris and OpenSolaris.

VxFS is also now sold by Symantec as a product for Windows, Solaris, and Linux called *Veritas Volume Manager (VVM or VxVM)*. *VxFS* provides Snapshot or as *VXFS* calls it mirroring of existing volumes for point-in-time backup purposes. The snapshot is taken using the mount command. Both the user and the backup utility have the same view of all data blocks in the filesystem. The snapshot uses a (smaller) volume for storing original blocks, should an update occur in the filesystem.

ZFS (Linux, Solaris and FreeBSD)

ZFS is an advanced file system and volume manager which was originally developed for Solaris. It has been successfully ported to multiple platforms and now there is a a functional *Linux ZFS* kernel port. The features of *ZFS* include data integrity verification against data corruption modes, support for high storage capacities, integration of the concepts of filesystem and volume management, snapshots and copy-on-write clones, continuous integrity checking and automatic repair.

NetApp® core Snapshot®

NetApp® core Snapshot® and disk-to-disk replication technologies allow point-in-time backup copies in native format. *Core Snapshot* guarantees fast backups and recovery: native format enables end user browse and restore.

The technology allows to combine snapshots, copies, replication and tape in a single solution. It reduces disk capacity requirements of a strong percentage.

NetApp Snapshot makes efficient use of storage by storing only block-level changes between the creation of each successive snapshot copy. Because the snapshot process is virtually instantaneous, backups are fast and simple. Snapshot copies can be automatically scheduled, they can be called from a script running on a server.

Microsoft Volume Snapshot Service (VSS)

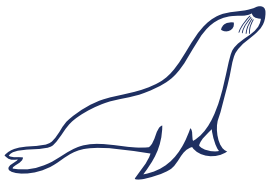
VSS is a technology included in Microsoft Windows that allows taking manual or automatic backup copies or snapshots of data, even if it has a lock, on a specific volume at a specific point in time over regular intervals. It is implemented as a Windows service called the *Volume Shadow Copy* service. A software *VSS provider* service is also included as part of Windows to be used by Windows applications. *Shadow Copy* technology requires the file system to be *NTFS* to be able to create and store shadow copies. Shadow Copies can be created on local and external (removable or network) volumes by any Windows component that uses this technology, such as when creating a scheduled Windows Backup or automatic System Restore point.
Applications can provide specific support for VSS through VSS writers which control how data is set to a consistent state at the beginning of a VSS operation and maintain that consistency throughout the process, among other functions.

R1Soft Continuous Data Protection® (CDP)

CDP is an online backup. It takes Snapshots of InnoDB and MyISAM tables in less than a second for online backups with zero interruption to users.
CDP is a block-based backup technology that bypasses files and file systems almost completely and read data directly form the Disk or Volume.
A big advantage of bypassing the file system is that there is no penalty on backup performance for having a large number of files. The backup application never looks at files and does not care how many there are. It also reads blocks in the order that they are on the disk, not the order that they appear in files which tends to be heavily fragmented and causes the disk(s) to spend more time seeking than actually reading data.

The most efficient method for computing deltas is the near-Continuous or *CDP* method. R1Soft happens to be the only example of a near-Continuous deltas method for both Windows and Linux platforms. The near-Continuous method works by using a disk volume device driver. The device driver is situated between the file system (e.g. NTFS) and the disk volume (e.g. Logical Disk Volume 1). *Continuous Data Protection* is not concerned with the number of files. It does not index file on the file system. It only reads deltas. It knows what the deltas are before the backup operation starts because of the near-continuous method for computing deltas that are simply defined as the data that has changed since the last backup run.

R1Soft's proprietary Linux device driver (it relies on *VSS* on Windows) tracks low level block deltas as they happen to minimize backup windows down to only reading deltas.
The *R1Soft CDP Server* uses proprietary *Disk Safe* technology to archive hourly, daily, weekly, or monthly disk-based MariaDB or MySQL backups. Only block level deltas are stored for every recovery point. Each recovery point appears as a full backup. This is called Virtual Full Backup.



How Does Continuous Data Protection® for MariaDB and MySQL Work on Linux?

These steps are the same for both MyISAM and InnoDB. If a database contains only InnoDB tables then the *FLUSH TABLES* and *READ LOCK* queries are only needed to synchronize the MariaDB or MySQL binary log if it is enabled..

1. Continuous Data Protection Server (CDP Server) connects to CDP Agent on target MariaDB or MySQL server to request backup operation (synchronization).
2. Agent Repeats the Following Steps *Three Times*:
   - execute FLUSH TABLES
   - sync disk containing MariaDB or MySQL data directory (uses proprietary Linux *IOCTL* call on R1Soft *CDP* device driver to sync only MariaDB or MySQL data directory disk)
3. Execute *FLUSH TABLES WITH READ LOCK.* After the previous step this typically takes less than 1 second
   (Note: unusually long running queries on a MariaDB or MySQL server can cause this step to take longer)
4. Create a Disk/Volume Snapshot using the proprietary *R1Soft Linux CDP Device Driver*. This typically takes less than 1 second.
5. *UNLOCK TABLES.* After this point MariaDB or MySQL queries can proceed again.
6. Read Block Level Deltas from the Snapshot and send them over the network to the CDP Server Disk Safe Repository
7. Remove the Snapshot

MariaDB and MySQL is only locked for steps 3) and 4). These steps combined typically take less than 1 second even on write intensive MariaDB and MySQL servers; Step 6) is where the *Continuous Data Protection* comes in. Using a proprietary low level Linux device driver the *R1Soft Linux Agent* knows in advance what block level changes where made to the MariaDB and MySQL data files for MyISAM and InnoDB. The backup window only consists of reading changed blocks from disk. Delta computation meaning determining what files or blocks have changed is done real-time as the system is running.

Advantages

One of the most important benefits of *CDP* is that only deltas are read during the backup operation. It works when MariaDB and MySQL files are located on different devices, both for Linux and Windows.

Rsync Backup

rsync is an efficient method for synchronizing a set of files over the network. *rsync* can compute the differences in two file trees efficiently using an ingenious specialized method of computing deltas using checksums called the *rsync algortithm*. Deltas are the parts of files that have changed from one version of a file set to another. *rsync* defines deltas as variable length blocks or chunks of data.

Using checksums to determine the differences between files is nothing new. What *rsync* added is the idea of a weak and strong checksum. For each file, *rsync* breaks the file into small fixed size chunks of data called blocks. Each block of all files are read into memory where *rsync* computes the weak checksum (adler-32) of the block. The adler-32 checksum requires very little CPU time to compute. The weak checksum of a new block in a file is compared to the weak checksum of a block in the older version of a file. If the weak checksum is different, the block has changed and is copied from the source to the destination.

rsync has been developed for general purpose file transfer and mirroring over slow WAN links.

Advantages
The whole idea of the rsync algorithm is to copy as little data across a slow network link as possible to update an older version of a set of files on another computer. *rsync* does a great job of this as it's guaranteed to only send deltas or different parts of files that have actually changed.
In doing this, *rsync* aims to reduce CPU time needed to compute checksums.
Many scripts are available to allow incremental backups

Disadvantages
The real challenge as any server administrator knows is that in a modern server CPU time is usually plentiful while disk I/O is very precious. rsync must read all of the data on disk or in the file set to compute the checksums. This means the *rsync* operation takes longer and longer the more data there is no matter how little data has actually changed.

Another challenge is that *rsync* works at the level of files and directories like any other program. This might seem confusing considering the previous paragraphs discussed how *rsync* divides files into blocks. What this means is that *rsync* examines each file and must create an index of the path to every file and directory on the server. For a server with a fairly large number of files this can be an enormous task all on its own. It's also the reason people often have problems with rsync's consumption of large amounts of memory on larger servers. rsync is not able to get a consistent copy of data

Oracle® InnoDB Hot Backup (*ibbackup* and *innobackup*)

The *ibbackup* utility provides an online backup capability for InnoDB table spaces. It means zero business interruption during backups for InnoDB. It is a hot backup: it is possible to run *ibbackup* without shutting off or even locking tables during the backup. *ibbackup* works by reading the InnoDB journal log file and table space at a very low level to provide a point-in-time backup of the InnoDB table space while MariaDB and MySQL is writing to the table space.

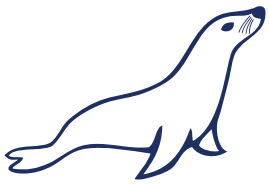
Innobackup is a Perl script that is an easy-to-use tool for making a complete backup of both MyISAM and InnoDB tables. It is used in combination with *ibbackup*, which is run as a child process by *innobackup*. In addition to creating backups, *innobackup* can prepare a backup for starting a MariaDB or MySQL server on the backup, and it can copy data, index, and log files from backup directory back to their original locations.

Both *ibbackup* and *innobackup* are now obsolete. They have been replaced by *mysqlbackup* from *MySQL® Enterprise Backup* that includes all the capabilities of *ibbackup* and *innobackup*.
ibbackup and *innobackup* commands are supplied as aliases or copies of the *mysqlbackup* command and backups produced by the InnoDB Hot Backup can be restored by the *MySQL® Enterprise Backup* product.

MySQL® Enterprise Backup (MEB - *mysqlbackup*)

MySQL Enterprise Backup is the newest version of the Oracle's InnoDB Hot Backup Utility and it is only available in selected commercial editions.

In terms of features, the MySQL Enterprise Backup product is a superset of InnoDB Hot Backup.



The MySQL Enterprise Backup product performs backup operations for MySQL data. It can back up all kinds of MariaDB and MySQL tables. It can relay on special optimizations for backups of InnoDB tables. MEB backups running databases without locking InnoDB data, can create compressed backups, partial backups and allows point-in-time recovery.

Recent MEB releases introduced interesting features like the backup of additional files, such as partition files, the backup of compressed tables and even the backup of in-memory database.
With MEB 3.8 it has been introduced an improvement in backup operation performance by implementing parallel processing of related internal operations; this means that some operations like reading and writing data or compression-decompression are performed in parallel taking advantage of multi-threading.

MySQL Enterprise Backup primarily operates through with the *mysqlbackup* command. This command has several options and it can perform different types of backup and restore operations.
The *mysqlbackup* command is a C program that connects to the server through the MySQL API, rather than a Perl script that runs the mysql command. Because it does not run the actual mysql command, it does not support the *--mysql-extra-args* option of the *innobackup*, but otherwise the syntax is compatible.

The *mysqlbackup* command is now a cross-platform replacement for the *innobackup* command, and it is available on Windows (Windows users can back up tables from other storage engines besides InnoDB, such as MyISAM tables, without writing their own wrapper scripts).

The optimization within the *ibbackup* command that skipped copying unused space within InnoDB tablespace files, is available within *mysqlbackup* only in combination with the *--compressed* option.
Currently, the old ibbackup and *innobackup* commands are still supplied as aliases or copies of the *mysqlbackup* command. When mysqlbackup is executed under these names, it accepts the same old option syntax from those commands. This backward compatibility is for troubleshooting in case of upgrade issues in the transition to the mysqlbackup command.

All backup-related operations either create new files or reference existing files underneath a specified directory that holds backup data. This directory must be on a file system with sufficient storage. (It could even be remotely mounted from a different server.)

mysqlbackup can perform several kinds of backup, restore, and pack/unpack operations.
One of the most interesting features is the backup-and-apply-log option that performs an extra stage after the initial backup to bring all InnoDB tables up-to-date with any changes that occurred during the backup operation, so the backup is immediately ready to be restored.

For backups of large or busy databases, though, it might be convenient to split up these stages to minimize load on the database server. That is, *mysqlbackup* can be first executed with the `backup` option, the backup can be transferred to another server, then the *mysqlbackup* command with the `apply-log` option should be executed to perform the final processing.

mysqlbackup records a unique ID for each backup job in special tables that it creates inside the instance: this allows to monitor long running backups and view the results of previous backups and it is possible

to specify a Log Sequence Number (*LSN*) to represent the point in time from which to take an incremental backup. The relevant *LSN* is displayed by the output of the *mysqlbackup* command. Once the LSN corresponding to the time of a full backup is known, it is possible to specify that value to take a subsequent incremental backup, whose output contains another *LSN* for the next incremental backup. *mysqlbackup* can also verify a backup before restoring it.

Advantages

1. InnoDB and MyISAM tables are supported
2. Windows
3. C program
4. Logging capability that records progress of running job
5. Historical details for completed backup jobs
6. Supports incremental backups
7. Single table backup available
8. Incremental
9. Consistent
10. Local or remote (streaming available)

Disadvantages

1. It only comes with enterprise edition of MySQL® server and under subscription

Percona XtraBackup

It can be defined as the open source version of InnoDB® Hot Backup tool (now included in MySQL® Enterprise Backup). It has a strong support from Percona and from MariaDB, from the Community and key users.

XtraBackup is the world's only open-source, free MariaDB and MySQL hot backup software that performs non-blocking backups for InnoDB and XtraDB databases.
XtraBackup is a combination of the xtrabackup C program, and the *innobackupex* Perl script. The *xtrabackup* program copies and manipulates InnoDB and XtraDB data files, and the Perl script enables enhanced functionality, such as interacting with a running MariaDB or MySQL server and backing up MyISAM tables. XtraBackup works with unmodified MariaDB or MySQL servers, as well as Percona Server with XtraDB. It runs on Linux and FreeBSD. MariaDB has ported Xtrabackup on Windows and it is now available for production.

How does XtraBackup work?

XtraBackup is based on InnoDB's crash-recovery functionality. It copies your InnoDB data files, which results in data that is internally inconsistent; but then it performs crash recovery on the files to make them a consistent, usable database again.

This works because InnoDB maintains a redo log, also called the transaction log. This contains a record of every change to InnoDB's data. When InnoDB starts, it inspects the data files and the transaction log, and performs two steps. It applies committed transaction log entries to the data files, and it performs an undo operation on any transactions that modified data but did not commit.

XtraBackup works by remembering the log sequence number (LSN) when it starts, and then copying away the data files. It takes some time to do this, so if the files are changing, then they reflect the state of the database at different points in time. At the same time, *XtraBackup* runs a background process that watches the transaction log files, and copies changes from it. *XtraBackup* needs to do this continually because the transaction logs are written in a round-robin fashion, and can be reused after a while. *XtraBackup* needs the transaction log records for every change to the data files since it began execution.
The above is the backup process. Next is the prepare process. During this step, *XtraBackup* performs crash recovery against the copied data files, using the copied transaction log file. After this is done, the database is ready to restore and use.

The *innobackupex* program adds more convenience and functionality by also permitting you to back up MyISAM tables and .frm files. It starts *xtrabackup*, waits until it finishes copying files, and then issues *FLUSH TABLES WITH READ LOCK* to prevent further changes to MySQL's data and flush all MyISAM tables to disk. It holds this lock, copies the MyISAM files, and then releases the lock.

The backed-up MyISAM and InnoDB tables will eventually be consistent with each other, because after the prepare (recovery) process, InnoDB's data is rolled forward to the point at which the backup completed, not rolled back to the point at which it started. This point in time matches where the *FLUSH TABLES WITH READ LOCK* was taken, so the MyISAM data and the prepared InnoDB data are in sync.
In other words, with XtraBackup, the actual point-in-time is a moving target until the backup process is complete. For example, if XtraBackup starts at midnight and lasts till 2:20 AM, then the backup's actual point-in-time is 2:20 AM.

The **xtrabackup** and **innobackupex** tools permit to do operations such as streaming and incremental backups with various combinations of copying the data files, copying the log files, and applying the logs to the data.

Advantages

1. Open source and free
2. Backups that complete quickly and reliably
3. Uninterrupted transaction processing during backups
4. Savings on disk space and network bandwidth
5. Automatic backup verification
6. Higher uptime due to faster restore time
7. Supports InnoDB, XtraDB, and MyISAM (table lock at the end of the backup), Archive, Partitioned tables, triggers and options
8. Supports all versions of Percona Server, MariaDB, MySQL and Drizzle (in Beta).
9. It performs streaming, compressed, and incremental MariaDB or MySQL backups.

Disadvantages

1. Backups may be bigger that the data that was copied.
2. Having to run XtraBackup more than one time. First, to copy the data. Next, copy missing transactions for the time the backup was done until its first completion. Finally, to create

ib_logfiles for drop-in replacement of the current ib_logfiles. In some rare cases, if data is coming into InnoDB infrastructure faster than XtraBackup can copy, it runs in a catchup mode for data 90% of initial innobackupex phase. Sometimes, that full catchup never happens.

MySQL Data Dumper (mydumper)

My dumper is a high-performance, multi-threaded backup (and restore) toolset for MariaDB, MySQL and Drizzle. The main developers originally worked as Support Engineers at MySQL (and this is how they would envisage *mysqldump* based on years of user feedback).

mydumper is a tool used for backing up MariaDB and MySQL database servers faster than *mysqldump*. It also has the capability to retrieve the binary logs from the remote server at the same time as the dump itself. The advantages of mydumper are therefore parallelism and performance.

Mydumper maintains snapshots across all threads, and provides accurate master and slave log positions. It has a 'daemon' mode which lets it run in the background. When in daemon mode, *mydumper* takes multi-threaded consistent snapshots on a regular basis (60 minutes by default) whilst also pretending to be a MariaDB or MySQL slave and continuously retrieving a local copy of binary logs. This can, in-theory, give an up-to-the-second backup of the server.

For example, it is possible to make snapshots every 2 hours and constantly retrieve the binary logs. A maximum of 2 snapshots are retained, one for the previous snapshot and one for the current one in case a failure happens during the current snapshot. A symbolic link inside the dump directory will always point to the last good snapshot. Each snapshot contains the binary log positions from that snapshot so at any given time the snapshot can be loaded in using myloader (a multi-threaded restoration tool bundled with mydumper) and then the log applied from that point using *mysqlbinlog*.

Advantages

1. Lightweight C source
2. Up to 10x faster dumps compared to *mysqldump*
3. Consistent snapshots for transactional and non-transactional tables (in 0.2.2 onwards)
4. File compression on-the-fly
5. Binary log dumps
6. Multi-threaded restore utility (in 0.2.1 onwards)
7. Daemon mode for timed snapshots and continuous binary logs (in 0.5.0 onwards)
8. Open Source! (GNU GPLv3)

Mydumper is still under active development but is well tested/used in production on some large installations.

Other Tools

There are several tools and solutions available, mainly in the Linux environment, most of them are script that automate actions around standard backup tools, other are special features included in more complex tools that in some way can be used as a backup solution.

Maatkit/Percona toolkit (mk-parallel-dump and mk-parallel-import)

Maatkit is a powerful toolkit for working with MariaDB and MySQL created by Baron Schwartz from Percona, that now is known as Percona Toolkit.

mk-parallel-dump is a tool to dump MariaDB and MySQL tables in parallel. It's especially interesting for large databases that take a lot of time to backup with *mysqldump*.

AutoMySQLBackup

AutoMySQLBackup is a script designed to take daily, weekly and monthly backups of MariaDB and MySQL databases using *mysqldump*. It runs on Linux systems only.

The main features of AutoMySQLBackup are:

1. Backup all databases to a single backup file or to a seperate directory and file for each database.
2. Automatically compress the backup files to save disk space using either gzip or bzip2 compression.
3. Can backup remote MariaDB and MySQL servers to a central server.
4. Runs automatically using cron or can be run manually.
5. Can e-mail the backup log to any specified e-mail address instead of "root". (Great for hosted websites and databases).
6. Can email the compressed database backup files to the specified email address.
7. Can specify maximun size backup to email.
8. Can be set to run PRE and POST backup commands.
9. Choose which day of the week to run weekly backups.

The *AutoMySQLBackup* script only requires *mysqldump* and *gzip* or *bzip2* for compression of the backup files. The script can automatically rotate daily backups and create weekly or monthly folders.

Backup Manager and Tools

Backup Manager tools are third party products that share a common behavior: they do not add features to the current backup, they organize, optimize and use what it is available at OS or DB level. Some of these provide an agent to interact with MySQL®.

Acronis

Main features

- Full system recovery
- Integrated data protection
  - Integrated data protection gives a data-centric view for granular access to specific data within backups. As a result, you get the benefits of data and system bare-metal recovery in one product with a single-pass backup.

- Virtual environment support
  - High-speed, simultaneous, agent-less image-based backups and recoveries for virtual machines are managed from the same console and with the same instruments as physical machines. One Virtual

Edition license gives you backup and recovery of an unlimited number of virtual machines on a host, plus V2P and P2V migrations.

- Backup and recovery to cloud storage

- One unified solution
  - All in one powerful solution you can benefit from image-based system recovery for physical and virtual environments, backup for files and applications, search and catalog, as well as disk, tape and cloud-storage options.

Acronis supports Windows or Linux, physical and virtual machines running on VMware® vSphere/ESX/ESXi, Microsoft® Hyper-V™, Citrix® XenServer®, Red Hat® Enterprise Virtualization as well as Parallels® Server 4 Bare Metal.

Arkeia

Just as Arkeia Backup Servers manage the backup destination (e.g. disk, VTL, tape devices), Arkeia Backup Agents manage the source data to be backed up or restored.

Arkeia offers Backup Agents for over 200 platforms, including virtually all Linux and Windows platforms, as well as AIX, BSD, HP-UX, Irix, Macintosh, Netware, and Solaris. Typically, a single Backup Server will manage between a handful and a thousand Backup Agents, depending on variables including the file volume to be backed up, the network performance, and the length of the backup window.

The Arkeia Backup Agent Encryption Option allows source data to be encrypted on the source computer, before it is transferred over the network. Encryption at the source distributes the load across Backup Agents and ensures that data are secure during network transit.

The Arkeia Disaster Recovery Option allows a system image to be restored to a bare-metal computer. Arkeia support Disaster Recovery on the Windows and Linux platforms. A bootable CD is written on a Backup Server and used to re-image the target machine.

CA ARCServer

CA ARCserve® Backup supports cloud storage, it is possible to migrate backup data to either a public or a private cloud with ease. CA ARCserve Backup lets specify the cloud device in the final destination tab when the user enables staging from the CA ARCserve Backup Manager. CA ARCserve Backup provides the user with various tools and options that you can use to manage cloud storage. It is possible to create a virtual cloud-based device using CA ARCserve backup that enables CA ARCserve Backup to store data with the cloud vendor specified by the CA ARCserve Backup cloud connection. CA ARCserve Backup provides the user with dashboard reports that help him manage cloud storage in an efficient way.

Zmanda Recovery Manager

(Zmanda Recovery Manager for MySQL -mysql-zrm)

Zmanda Recovery Manager (ZRM) lets the user create full logical or raw backups of databases, generate reports about the backups, verify the integrity of the backups, and recover databases. It can

also send email notifications about the backup status, and let the user implement multiple backup policies.

How Does Zmanda Recovery Manager (ZRM) Actually Work?

ZRM is a perl-based web utility used to manage the open source Amanda application, using a variety of methods to extract data from MariaDB and MySQL and configuring scripts at backup time. ZRM adds a GUI to all of the Legacy MariaDB and MySQL backup techniques and provides online backup using Online Volume Snapshots (e.g. LVM).

Furthermore, ZRM provides integration between ZRM, underlying storage and MariaDB and MySQL binary logs enables to perform highly efficient Coninuous Data Protection (CDP) with almost instantaneous point-in-time recovery.

ZRM perform hot backup of live MariaDB and MySQL through:

- Logical backups (mysqldump)
- Physical Backups (mysqlhotcopy)
- Replication
- Storage snapshots support for full backups
  - Linux LVM
  - Microsoft VSS
  - Solaris ZFS
  - Amazon EC2 EBS Snapshot
  - Symantec Veritas VxFS(*)
  - NetApp SnapManager(*)
  - EMC SnapView technologies (*)
    (*)*Symantec VxFS, NetApp SnapManager, EMC SnapView technologies are available as Add-on Options*.

- XtraBackup Support for Non-Blocking Backups
  - ZRM with XtraBackup provides resource utilization management by providing throttling based on the number of IO operations per second. XtraBackup based backups also allow for table level recovery even though the backup was done at the database level (needs the recovery database server to be Percona Server with XtraDB).

- MySQL Enterprise backup

- Incremental Backups (based on MariaDB and MySQL Binary Logs)
  - This allows very frequent backups (e.g. hourly) as well as do a point-in-time restore between any two backups. No read or write lock is taken during incremental backups

ZRM provides unified point-in-time recovery for any type of backup method used

- Flexible recovery level,
  - recover the whole MariaDB or MySQL server, database, table or transaction
- Identify point-in-time and initiate the recovery through Visual Log Analyzer

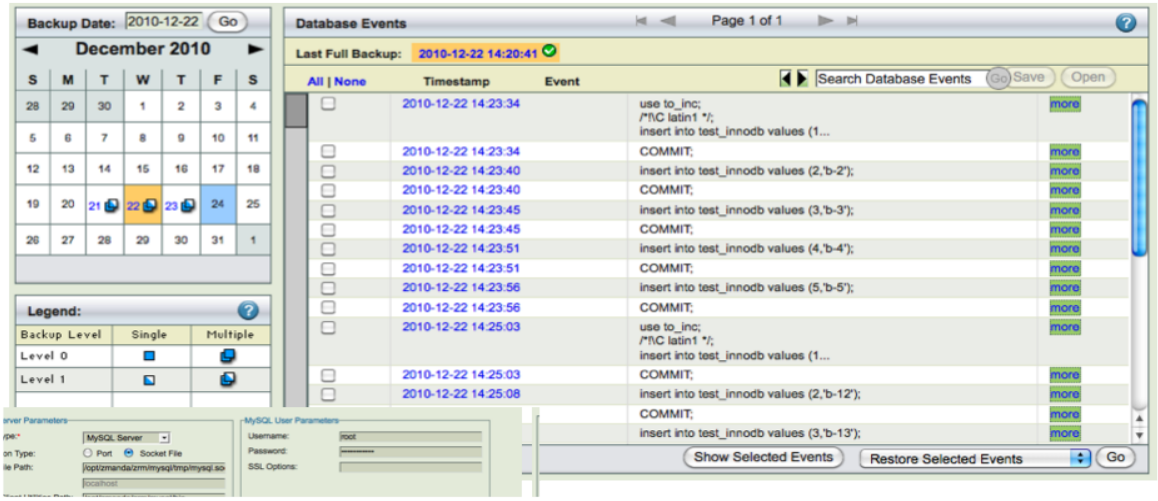
Centralized Global Backup Management (see picture)

- Automate backups across local, remote or clustered MariaDB and MySQL servers via an easy-to-use Web console
- Backup MariaDB and MySQL servers with MyISAM, InnoDB, NDB and other storage engines on Linux, Solaris, OpenSolaris, Windows and Mac OS X

- Backup Configuration files for Applications
- Easily set up backup & retention policies

ZRM has a multiplatform support

- Linux
- Solaris
- Windows (only up to ZRM 3.1)



Fast Backups of MariaDB or MySQL Running on Amazon EC2

- If MariaDB or MySQL databases run on the Amazon EC2 compute cloud, ZRM can perform fast full backups of these databases by using Elastic Block Store (EBS) Snapshots. ZRM takes only a momentary read lock on the MariaDB or MySQL database during the creation of the snapshot, in order to ensure consistency of the backed up database archive and only the blocks that have changed since the last full backup (via EBS snapshot) will be saved.
- ZRM automatically deletes EBS snapshots (containing full backups of MariaDB and MySQL) according to the configured retention policy. Just like other snapshot based full backups, ZRM intelligently correlates EBS Snapshots with incremental backups using MariaDB and MySQL logs, enabling the database administrator to recover their MariaDB and MySQL instances running on EC2 to any point in time.

Maximize Data Security

- Encrypt data-at-rest and data-in-transit using industry standard encryption technologies
- Configure on-the-fly compression to meet your storage needs
- Role-based access control provide added layers of security

Automated Alerts & Reports

- Built-in monitoring & reporting functions
- At-a-glance calendar view of all backup and recovery jobs
- Receive real-time email and RSS notifications on the status of your backups
- Easily fulfill audit or compliance requirements using highly configurable reports

Easily Deploy Your Backup & Recovery Solution

- ZRM is quick and easy to install
- Intuitive configuration with context sensitive help

ZRM does not use any proprietary format and this means that it is possible to restore MariaDB and MySQL backups without having ZRM installed on the server.

ZRM can do an Online backup of MariaDB and MySQL using Linux LVM, Veritas File System Snapshots, and the snapshot capability of Network Attached STorage devices like NetAPP and EMC. Zmanda can copy MariaDB and MySQL binary logs or provide a GUI for ibbackup (sold separately by Innobase).

Advantages

ZRM allows the use to manage backups accordingly with all the needs and configuration: ZRM makes it easy to recover data from incremental backups even DBAs have to use multiple incremental backup images to get data back to a particular point in time

Transportability of logical ZRM backup images makes ZRM a convenient tool for migration. For example, you can move your MariaDB and MySQL data:

- From MariaDB and MySQL on Solaris to MariaDB and MySQL on Linux
- From one storage engine to another
- From a 32-bit server to a 64-bit server
- From one managed hosting provider to your data center or another provider with different MariaDB or MySQL configuration

Both physical and logical backups provide warm backup, meaning it is necessary to shut down MariaDB and MySQL server for backup, but all tables are locked during backup and users can't enter their data.

One of the important considerations for remote backup of MariaDB and MySQL is the decision about what type of connection to establish between ZRM and remote MariaDB and MySQL server. ZRM provides a plug-in for socket based connection and another plug-in for SSH-based connection.

The socket copy plug-in is not secure and should be used only when security is not of concern or when security is established by other means, for example, when there is a VPN connection between ZRM and a remote MariaDB or MySQL server.

The SSH copy plug-in provides a secure channel between ZRM and the remote MariaDB and MySQL server. It uses public-key cryptography to authenticate the remote MariaDB or MySQL server and the backup user running ZRM.

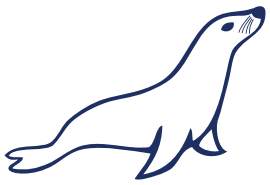
Disadvantages

Even though ZRM client can run both on Linux and Windows, the last one does not support Xtrabackup and it requires a specific Perl installation.

Conclusions

With all its rich functionality, ZRM for MariaDB and MySQL is just a tool for implementing backup and recovery strategy that is optimal for your specific data protection needs. ZRM is robust and easy to use, but depending on each implementation of the remote MariaDB or MySQL server and each specific requirements for backup and recovery, DBAs should consider all trade-offs associated with each

operational options provided by ZRM for MariaDB and MySQL. It is also important to say that flexible architecture of ZRM allows users to write their own plug-ins.



Evaluating a Backup Solution

Now that all the backup characteristics and opportunities have been listed, it should be easier to evaluate which is the most suitable backup solution for an organization.

It is fairly obvious that cold backups are pretty much unusable for live systems that cannot afford down-time. It is important to quantify the cost of data being unavailable to better offset the cost (hardware and software) of the backup solutions. Once aware of these costs, the next step is to have a clear picture of what the MariaDB and MySQL data is and, what is really necessary to backup. In fact, a large amount of transient data does not need to be backed up. It is also important to define how often a backup should be performed and where it should be stored.

A good starting point is answering the checklist of the five Ws (and one H) questions:
Some questions may have obvious answers, but the complete backup scenario is surely clearer after all the Ws questions have been answered.

Who is going to take care of a backup?

Why is a backup needed?

What needs to be backed up?

- Data base content (all database schemas? All tables? Static data?...)
- Log files
- Configuration files
- Cron jobs
- Other objects...

When is the backup planned?

- Are there usage peaks?
- Can a backup be scheduled on a regular basis?
- …

Where is the backup to stored?

- On the database server?
  - Is there at least a separate file system or volume or hard disk drive?
- On a remote server?
  - Is it onsite or offsite?
  - Will a cloud storage be used? Is data encryption needed!
- Is it possible to use multiple locations and share risks?
- …

How will the backup be performed?

There are also some points to be aware of:

- Snapshots should not be confused with backups

  - A Disk Snapshot is an image of a data disks, at a certain point in time. The snapshot freezes the state of the data and it does not provide any information of what happened to the data before it has been taken. If, for some reason, the data is physically or logically corrupted or inconsistent, this corruption or inconsistency will be frozen in the snapshot. In general, previous snapshots are overridden and they

are not stored. The frequency of the snapshots is in general higher than a standard backup.

- Logs should not be confused with backups
  - Log files collect the list of every action that has been performed on a database. The use of logs for backup purpose is very common, mainly because it is a very simple procedure and also an easy way to recover all the activity that has been performed on a database.
    Log files are very easy to be backed up.
    The granularity of a log file makes it easy to select a range of actions starting and ending at a certain point and time. Unlike snapshots, when data is restored from logs it is possible to stop the recovery process before the a logical inconsistency. Logs have usually checksums and they do not restore physically corrupted data.
    Restoring a backup from a log file, on the other hand, can be inefficient. For example, if an application performs an update of the same record a million times, the restore procedure will update the record a million times instead of jumping at the very llast value for that record.

A mixed approach that combine a set of multiple snapshots (taken several times a day every defined time interval) and log files that allow to recover data consistency from the snapshot time up to the last action before disaster could be a good way to get the most from the two solutions.
It is important to say, though, that snapshots can require a convenient amount of storage and maybe some written procedure that allow to manage a circular buffer of snapshots. Questions or comments are welcome and encouraged at www.mariadb.com.