Lock, Stock and Backup: Data Guaranteed

Jervin Real

August 16, 2017

PERCONA

Agenda

1. Backup and Recovery Primer
2. Binary Log Backups
3. Logical Backups with mydumper
4. Physical Backups with xtrabackup
5. Leveraging Snapshots
6. "3-2-1" in the Cloud
7. Monitoring, Testing and Recovery
8. Encryption and Security

Backup and Recovery Primer

Why do we take backups?

- Absolute, hands down, no contest, necessity. Period.
- Backups are the insurance you almost always never use, recovery is the ultimate test of reliability

Pillars of good backup implementation

- Restore/recovery time objective
- Restore/recovery point objective
- Retention period
- Storage
- Storage engines
- Tools and environment

Backup and Recovery Primer

Backup types we will discuss today:

- Binary Logs
- Logical Backups with mydumper
- Physical Backups with xtrabackup
- Filesystem Snapshots

Binary Log Backups

Binary Log Backups

Its not just another type of backup ...

- `mysqlbinlog` as primary tool of choice, but you can write your own!
- Support from `mysqlbinlog` only became available from 5.6, but can be used <5.6 servers
- Complements other backup types for Point in Time Recovery

Binary Log Backups

Pros and Cons?

- There is nothing to discuss really, just Yes or No
- Hopefully, you choose Yes
- Binary log backups is uniquely important on its own.

Binary Log Backups

Why

- If point in time recovery is important
- If you need some limited auditing capabilities

Why Not

- If you have rolled out a custom solution that is faster in your use case i.e. snapshotting a big table based on timestamp

Binary Log Backups

Binary log streaming basic form:

```
mysqlbinlog \
  --read-from-remote-server --host={hostname} \
  --raw --stop-never mysql-bin.000001
```

- Connects to `{hostname}` similar replication slave
- Downloads and save binary log events in its binary form, continously
- Starting from `mysql-bin.000001`
- Saves to disk under the same name and series as the source

Logical Backups with mydumper

Distinct features of mydumper

- Parallel logical dump and restore (think row level)
- Works with Percona Server backup locks
- Regex based include/exclude
- Kill or abort on long running queries
- Wire compression (mysql > (protocol) mydumper > (stream) disk)
- Multiple levels of lock reduction
- `SAVEPOINT` (`--use-savepoints`) to reduce MDL locks
- Everything else (mostly) mysqldump does
- Open source and community maintained

Logical Backups with mydumper

Pros

- Per object backup/restore, highly flexible with regex
- Storage engine agnostic
- Highly compressible, dump only data or even snapshots based on time
- Parallelization means speed advantage

Cons

- No piping/streaming, direct to disk only - means encryption comes after

Logical Backups with mydumper

mydumper

1. `~/.my.cnf` / `--defaults-file`
2. ZLIB output compression
3. `--regex`
4. No HEX dumps, BLOBs could be unreadable
5. Linux only, dump over TCP is possible from Windows source
6. Manual progress monitoring is tricky

mysqlpump

1. `--login-path`
2. ZLIB or LZ4 output compression
3. SSL connections
4. `--include`/`--exclude`
5. `--hex-blob`
6. Linux/Windows
7. `--watch-progress`

Example: Default Percona Managed Services configuration

```
mydumper --outputdir=/path/to/storage --verbose=3 --host=source_server_ip \
   --port=3306 --kill-long-queries --chunk-file-size=5120 --build-empty-files
```

- Dumps all object from the source i.e. db, tables, routines, views
- One dump per table, schema dump is separate from actual data
- Creates dump files even if tables are empty
- Kills long running queries that could block FTWRL

```
parallel -j4 --no-notice gpg2 --encrypt --recipient key@percona.com {} \
   '&&' unlink {} ::: *.gz
```

- Encrypts the backup files in parallel and removes original copy when done

Physical Backups with xtrabackup

Physical Backups with xtrabackup

What is xtrabackup?

- Open source alternative to MySQL Enterprise Backup
- Consistent online backup tool, highly configurable to many workloads

Physical Backups with xtrabackup

Pros

- Typically faster, parallelization options available
- Least intrusive - multiple levels of locking granularities for consistency
- Per object backup possible
- Incremental backups available
- Compression and Encryption features

Cons

- Linux-only official support
- Occupies more disk space (compression ratio depends on state of data on disk)
- Designed for InnoDB data and workload

Example: Encryption with GPG

```
xtrabackup --backup --stream=xbstream --target=./ |
   \ gpg --encrypt --recipient key@percona.com > backups.xbs.gpg
```

- Better control of encryption keys, storage is not tied to encryption
- No parallelization via stream, backup and encrypt separately if needed (22mins vs 8mins)
- https://twindb.com/encrypting-mysql-backups/

Example: Stream backup to multiple destinations at once with FIFO

1. Last receiving server/slave

```
nc -l 9999 | xbstream -x -C /path/to/destination
```

2. Receiving server/slave 1-to-N

```
mkfifo xbfifo
*nc IP_of_next_receiving_server 9999 < xbfifo &
nc -l 9999 | tee xbfifo | xbstream -x -C /path/to/destination
```

3. Backup source

```
xtrabackup --stream=xbstream --backup --target=./ \
| nc IP_of_first_receiving_server 9999
```

Example: Backup to populate a Galera/PXC node manually

```
xtrabackup --backup --stream=xbstream --target=./ --galera-info \
   | nc IP_of_dest_galera_node 9999
```

- Make sure the data on the destination is cleared and ready to receive on `netcat` port 9999
- Backup will include xtrabackup_galera_info https://www.percona.com/blog/2012/08/02/avoiding-sst-when-adding-new-percona-xtradb-cluster-node/

Filesystem Snapshots

Filesystem Snapshots

Why consider snapshots:

- Mixing storage engines i.e. InnoDB + TokuDB
- Really large datasets, less frequency, must have at least N intervals
- When it is more practical solution i.e. GCE snapshots

Filesystem Snapshots

What is available commonly used?

- LVM
- ZFS/ZVOL
- Amazon EBS
- Google Cloud Engine

LVM Snapshots

```
mysql> FLUSH TABLES;                  -- optional, to just preflush without locking
mysql> FLUSH TABLES WITH READ LOCK;
shell> sudo sync
shell> lvcreate --snapshot --name mysql-nnnnnn /dev/vgname/lvname
mysql> UNLOCK TABLES;
shell> mount /dev/vgname/mysql-nnnnnn /tmp-vol
shell> rsync -avz /tmp-vol /path/to/storage
shell> umount /tmp-vol
shell> lvremove -f /dev/vgname/mysql-nnnnnn
```

- Use on non-critical nodes if possible, LVM snapshots has performance penalty
- Easy to write tools, `mylvmbackup` was once defacto tool

Filesystem Snapshots

ZFS/ZVOL Snapshots

```
mysql> FLUSH TABLES;                 -- optional if excl innodb and iflatc = 1
mysql> FLUSH TABLES WITH READ LOCK; -- or if you need binlog coordinates;
shell> zfs snapshot zpool/data@date
mysql> UNLOCK TABLES;               -- optional, see above
```

- Recommend only when running on native platform i.e. BSD/Solaris
- Have not seen production cases for ZFSonLinux

Filesystem Snapshots

EBS Snapshots

```
mysql> FLUSH TABLES; -- optional
mysql> FLUSH TABLES WITH READ LOCK;
shell> sudo sync
shell> sudo fsfreeze -f disk-name
shell> aws ec2 create-snapshot
shell> sudo fsfreeze -u disk-name
mysql> UNLOCK TABLES;
```

- Once the snapshot is taken, no additional step is required, uploads to S3 in the background

Filesystem Snapshots

GCE Snapshots

```
mysql> FLUSH TABLES; -- optional
mysql> FLUSH TABLES WITH READ LOCK;
shell> sudo sync
shell> sudo fsfreeze -f disk-name
shell> gcloud compute disks snapshot disk-name
shell> sudo fsfreeze -u disk-name
mysql> UNLOCK TABLES;
```

- Wait until snapshot is uploaded to GCS
- No performance impact on storage (as per Google doc)
- https://cloud.google.com/compute/docs/disks/create-snapshots

3-2-1 In The Cloud

3-2-1 In the Cloud

Its a cloud selling point (1/2):

- Cloud providers will have their own optimized backup solution for hosted MySQL service
- Varies by provider and even by service versions, read their documentation!
- Mostly based on storage snapshot except Oracle MySQL Cloud Service

3-2-1 In the Cloud

Its a cloud selling point (2/2):

- AWS RDS Snapshot
  - Storage based, closely tied to EBS snapshots but tailored to MySQL consistency
  - Use multi-AZ to avoid brief IO suspension on snapshot creation
  - Options for encryption at rest, default on transit from EBS to S3
- Google Cloud SQL
  - No FTWRL (think encapsulated)
  - Support PITR with binary logs!
  - Encryption everywhere

Monitoring and Testing

Monitoring and Testing

Really?

- >50% of the work involved to ensure objectives are met
- Test not only the backup but also the process and the humans interacting
- Allows you to tune your implementation based on evolving needs

What do we test?

- Ideally, any backup that ends up on disk should be verified
  - Full + incrementals are working properly
  - Attach restored instance as slave, run checksums for consistency
- Restores (including PITR) - backups are in ready/usable state from storage/backup server
- Recovery (including PITR) - service is restored from backup server to production instance

Monitoring and Testing

What do we monitor?

- Space available? Look before you leap!
- Backup times, including its phases along the way
- Backup end result, including states on different phases along the way (keep your logs)
- Resulting backup size overtime

Security and Encryption

Which parts are secure?

mysqlbinlog

- SSL connections, SSH tunneling or VPN in transit
- Stored binlogs encrypted post-rotate or fs encryption

mydumper

- SSL, SSH tunneling, VPN in transit
- Stored encrypted filesystem or GPG encrypted to offsite
- Encryption is post backup, unless storing on encrypted storage, make sure temporary storage is secured as well

Which parts are secure?

xtrabackup

- SSL connections on metadata queries
- Encryption on stream can add latency to backup times
- Further encryption on final storage or offsite and during transit

Snapshots (on premise)

- Local to local snapshot copy is as secure as source data
- Encryption generally matters when exporting snapshot data
- Filesystem encryption, GPG, SSH tunnels, VPN

So, which one is the BEST?

Questions!

Percona Live Europe Call for Papers is Open!

Championing Open Source Databases

- MySQL, MongoDB, Open Source Databases
- Time Series Databases, PostgreSQL, RocksDB
- Developers, Business/Case Studies, Operations
- September 25-27th, 2017
- Radisson Blu Royal Hotel, Dublin, Ireland

- Advanced Reg Tickets Available Until Sep 5th!
- Last chance to save €25, hurry before prices go up Sep 6th

https://www.percona.com/live/e17/percona-live-call-for-papers

PERCONA