Validating Your Backups

Strategies to ensure your backups work

PERCONA LIVE

Matthew Boehm

April 15 2015

Outline

- Why should you validate?
- Brief overview of common backup strategies
  - mysqldump
  - mydumper
  - Snapshots (LVM, SAN, EBS)
  - Percona XtraBackup
  - Using a Slave

Outline

- Safety Precautions
  - Offsite
  - Binlogs
- Validation Strategies
  - Using Binlogs
  - Checksums
  - Replication

About Me

- Matthew (not Matt) Boehm
- With Percona November 2012 / PayPal Before
- Percona MySQL Architect
- Percona North American Trainer
- National Advanced-Level West Coast Swing

Validating Your Backups

WHY SHOULD YOU VALIDATE?

Show of hands...

"You haven't successfully taken a backup until you have recovered using that backup." -Shlomi

Why Take Backups?

- Disaster Recovery
  - Power Outage
  - Firmware Bugs
  - Hardware Failure
  - Application-Induced Mistakes

Backups Can Fail!

- Actual process of backup can fail
- Backing up invalid data
- Backing up incomplete data
- Offsite storage of backup

Validating Your Backups

COMMON BACKUP STRATEGIES

mysqldump

- `$ mysqldump --all-databases --single-transaction --master-data=1 > backup-file.sql`
- Restore with `mysql < backup-file.sql`
- Full-Table Scans
- No Parallelism
- Slowest to restore; executing INSERT statements

mydumper

- Logical backup tool; “mysqldump reimagined”
- Multi-threaded; export and import
- On the fly compression
- Daemon mode for schedules and continuous binlog dumps
- https://launchpad.net/mydumper

Snapshots

- In all snapshot cases, MySQL $datadir and $innodb_home must be on same device
- FTWRL
- Crash recovery on restore

Snapshots

- LVM (mylvmbackup)
  - Perl wrapper script
  - Everything on same LVM with free snap space
  - Copy to offsite

Snapshots

- EBS / XFS
  - xfs_freeze / ec2_create_snapshot
  - Wrapper Script: *ec2-consistent-snapshot*
- SAN
  - API/CLI to SAN

Percona XtraBackup

- Hot Backups (InnoDB Only*)
- No FTWRL
- Parallelism / Encryption / Compression

Slave Backups

- Any of the previous strategies
- Possible cold backups
- Non-master impacting
- Ensure true replica with *pt-table-checksum*

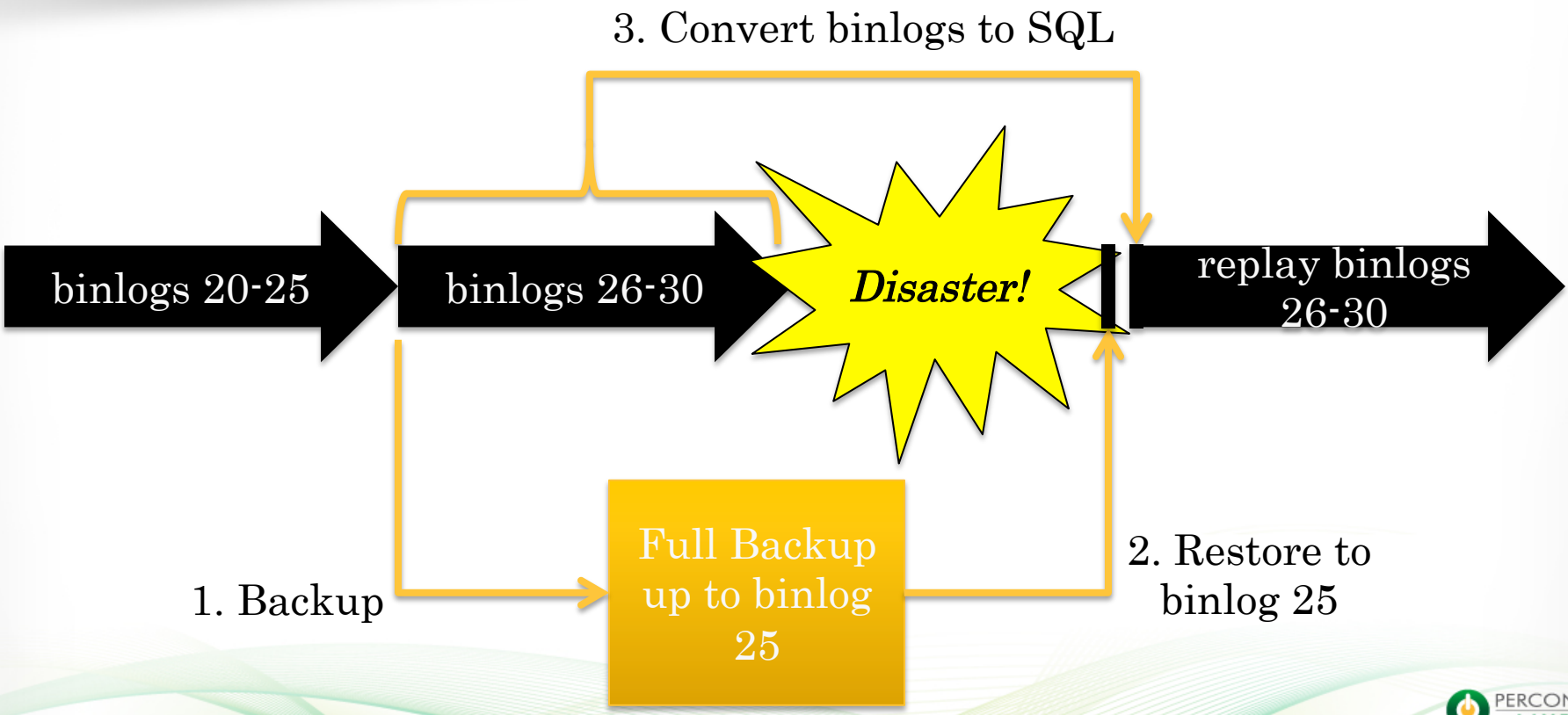
Validating Your Backups

SAFETY PRECAUTIONS

Safety Precautions

- Backup Location
  - Local Copy
  - Offsite Copy
- Binary Logs
  - rsync
  - mysqlbinlog

Binary Logs - PITR



Binary Logs

- Keep them:
  - *expire_logs_days = 0*
- Keep them safe:
  - cron'd rsync to remote server
  - `mysqlbinlog --read-from-remote-server --raw --stop-never`

Validating Your Backups

VALIDATION STRATEGIES

Validation Overview

- Restore to blank host
- Apply any/all strategies
  - Replication + Checksum
  - Binary Log Playback
  - CHECKSUM TABLE / Fragments
  - Smoke Tests / Unit Tests

Restore to Host

- Logical Backup
  - Shut down MySQL / Install Fresh
  - *rm -rf $datadir/**
  - *mysql_install_db*
  - *mysql < [zcat] /path/to/dump.sql[.gz]*
  - *myloader --directory=/path/to/dump --threads=4*

Restore to Host

- Physical Backup - Snapshots
  - Shut down MySQL
  - *rm -rf $datadir/**
  - Mount snapshot
  - cp -r /mnt/snapshot/* $datadir/
  - chown -R mysql:mysql $datadir

Restore to Host

- Physical Backup - Percona XtraBackup
  - Shut down MySQL / *rm -rf $datadir/**
  - innobackupex --apply-logs --use-memory=8G /path/to/backup/
  - innobackupex --copy-back /path/to/backup/
  - chown -R mysql:mysql $datadir

Strategy 1 - Replication + Checksum

- Configure restored host as slave of master
- Host should immediately resume replication
- Allow slave to catch up
- Use *pt-table-checksum* on master to verify all rows are intact
- If differences, check other slaves

Strategy 2 - Binary Log Playback

- Copy / Retrieve N binary logs from source host
- *mysqlbinlog mysql-bin.0034**

  -j <posOfBackup> | mysql
- Works best in ROW-based binlog format

Strategy 3A - CHECKSUM TABLE

- *CHECKSUM TABLE foo;*
- Requires that you collected checksums on the source instance while the database was locked.
- Even without comparison, confirms that you can read every row in the restored database.

Strategy 3B - Checksum Fragments

- Used on data that doesn't change after being INSERT'd

```
mysql -BNe "SELECT col1, col2, col3 FROM
foo WHERE col1 < 1000000
ORDER BY col1"| md5sum
```

Strategy 4 - Smoke Tests / Unit Tests

- Configure dev-application to access restored host
- Run pre-defined tests; Verify operation
- Verify row counts based on established values

Validating Your Backups

FINAL TIPS

Final Tips

- Checksum the backup / dump
  - SHA1 / MD5
- After restore/validation, checksum of dump provides ongoing validation

Validating Your Backup

Q&A

Thank You!

Thank You For Attending!

http://www.percona.com/training