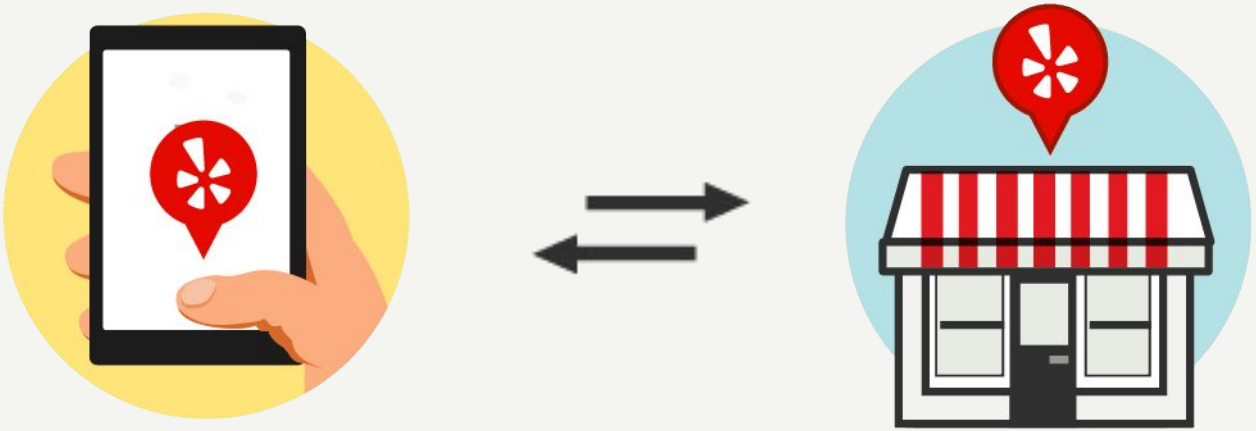
The Accidental DBA

Jenni Snyder
Yelp
@jcsuperstar
http://bit.ly/the_accidental_dba

Yelp's Mission

Connecting people with great local businesses.



Administrative

9:30am: Welcome & Let's talk about the basics
11:00am: Break
11:15am: Running MySQL in the real world
12:15pm: Final wrap up
12:30pm: Lunch

Agenda in More Detail

Part 1:

0. Brief Introduction
1. MySQL History, Philosophy, Community
2. Installation, Basic Configuration, Launching
3. Using it, Backing it up
4. Replication basics, Monitoring
5. Database Defense & fast feedback to developers

Agenda in More Detail

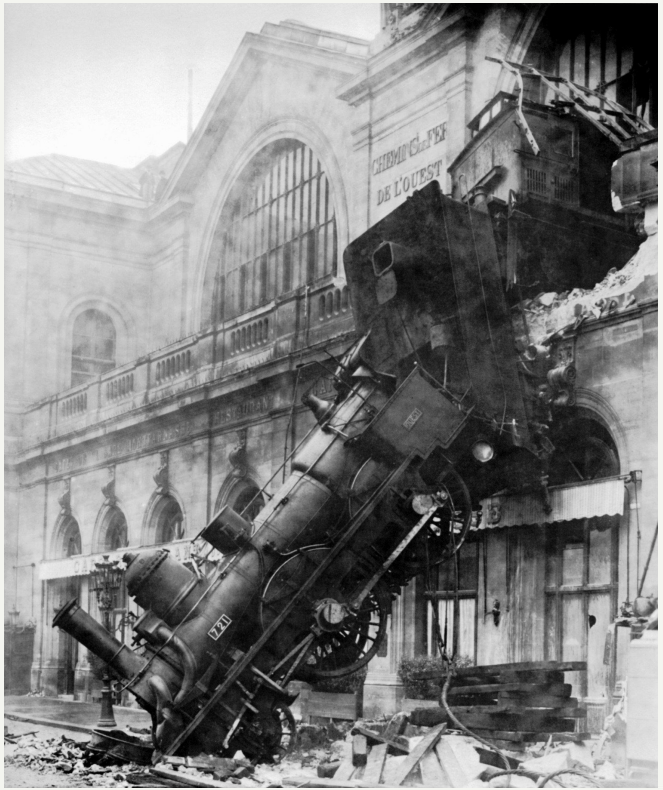
Part 2:

- Running in Production #realtalk
- Being Prepared
- Advanced Monitoring and Investigation
- Fantastic Errors and Where to Find Them
- When Things Are "Slow"
- Advanced Tuning
- Replication and Scaling Out
- Discussion

Who am I Why am I Here?

- My name is Jenni
- Engineering Manager @ Yelp
- First DBA there in 2011

Got Into Databases by Accident



Myth 1: *databases are scary!*

Myth 2: *DBAs are scary!*

Why Databases?

It Was on Fire When I Got Here

You may already have a database.

And it's Not Clear Who Fixes it

"*If someone is asking you who owns the database, it's probably you*"

- @mipsytipsy

Don't Feel Bad!

- Yelp didn't have a DBA for >5 years
- We've acquired companies with great big DBs
- There are probably some easy things to fix


Oat and Raisin Cookie
A classic cookie with rolled oats and succulent raisins.

Just in Case You Don't Have a DB Yet...

Database Flavors:

- SQL
- NoSQL
- Geospatial
- Graph

SQL/Relational Databases

- MySQL
- PostgreSQL
- SQLite

NoSQL

- Cassandra
- MongoDB
- Couchbase
- Redis

Geospatial & Graph

- PostGIS
- Neo4j
- Titan

When Not to Use a Database

When it's a:

- log
- queue
- cache
- photo
- disk directory
- spreadsheet
- data warehouse/cube/shed/lake/garage/whatever

Use the right tool for the job.


3
2
12
1
4
11
10
8
6
5
7
9

Brief History of MySQL

- MySQL - Wikipedia
- First released in 1995
- Pluggable storage engines
- Written in the age of spinning rust
- Transactional engine **InnoDB** released in 2001
- Oracle buys InnoDB in 2005
- Bought by Sun in 2008
- Oracle bought Sun in 2010

MySQL - Why Does it Do What It Does

- It is meant to work best the most of the time
- It is ~~hard~~ impossible to optimize for all cases
- That sometimes surprises you

"*Performs extremely well in the average case" - Wikipedia*

The MySQL Community

- The manual is awesome
- DBA StackExchange
- So many forums - "just <search engine> for it"
- Conferences
- Lots of great open source

Highly Configurable

- MySQL Manual - Server System Variables - 500+
- 400+ status metrics

Don't freak out. You don't have to care about these yet :)

Installation

- Source, package, #whatever
- Drop a config in `/etc/my.cnf`:
  - basedir, datadir
  - socket
  - pid-file
  - **server-id**

MySQL Files - Logs

- Error log
- Slow log
- Don't log warnings for now

MySQL Files - Replication Logs

- You may have lots of these:
  - Binary logs
  - Relay logs

MySQL Config - memory

- innodb_buffer_pool_size ~ 75% memory
- Worry about the rest of it later

Let's Fire This Puppy Up

- `service mysql start`

or

- `/etc/init.d/mysql` start
- Runs on port 3306
- Throw a party
- Or just connect to it
  - Use the `mysql` command

```
jsnyder@db:~$ ps auxw| fgrep mysql

root     16661  0.0  0.0   4460  1692 ?        Ss   Oct09   0:00
/bin/sh /usr/bin/mysqld_safe

mysql    17877  1.2 88.8 30221480 27893084 ?   Sl   Oct09 318:00
/usr/sbin/mysqld --basedir=/usr
--datadir=/nail/databases/mysql/data
--plugin-dir=/usr/lib/mysql/plugin --user=mysql
--log-error=/nail/databases/mysql/log/err.log
--pid-file=/nail/databases/mysql/mysqld.pid
--socket=/nail/databases/mysql/mysqld.sock --port=3306
```

```
jsnyder@db:~$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1124438
Server version: 5.6.32-78.1-log Percona Server (GPL), Yelp Release 1

Copyright (c) 2009-2017 Percona LLC and/or its affiliates
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

```
mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.02 sec)
```

When it Won't Start

- Find the start script
- Find that err.log
- Run `mysqld_safe` manually

MySQL Grants

- MySQL uses `GRANTS`
- Users are a combination of:
  - User (name)
  - Password
  - Host - % is wildcard
- `localhost` uses unix domain socket
- `127.0.0.1` uses TCP/IP

MySQL Grant Levels

- `GRANTS` can be for different levels:
  - Global `*.*`
  - Database `yelp_db.*`
  - Table `yelp_db.user`

```
mysql> CREATE USER 'yolo'@'%' IDENTIFIED BY '' ;
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL ON *.* TO 'yolo'@'%' WITH GRANT OPTION ;
Query OK, 0 rows affected (0.00 sec)


mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON first_db.* TO
'app'@'10.10.10.%' IDENTIFIED by 'sweetpassword' ;
Query OK, 0 rows affected (0.00 sec)


mysql> GRANT CREATE, DROP, INDEX, TRIGGER, SELECT, INSERT,
UPDATE, DELETE ON *.* TO 'schema_manager'@'10.%' IDENTIFIED BY
'yougettheidea' ;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>  select user, host from mysql.user ;
+----------------+------------+
| user           | host       |
+----------------+------------+
| yolo           | %          |
| schema_manager | 10.%       |
| app            | 10.10.10.% |
| mysql.sys      | localhost  |
| root           | localhost  |
+----------------+------------+
5 rows in set (0.00 sec)
```

```
mysql> show grants for 'root'@'%';
+-----------------------------------------------------------+
| Grants for root@%                                         |
+-----------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'%' WITH GRANT OPTION      |
+-----------------------------------------------------------+
2 rows in set (0.00 sec)


mysql> show grants for current_user ;
+-----------------------------------------------------------+
| Grants for root@%                                         |
+-----------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'%' WITH GRANT OPTION      |
+-----------------------------------------------------------+
2 rows in set (0.00 sec)
```

Debugging Connection Issues

- Make sure MySQL is running
- On that host & port
- Match user & host in error with grants
- Read `err.log`
- Use `skip-grant-tables` to start over

What if You Don't Know the Password?

- Add `skip-grant-tables` to your `my.cnf`
- Restart MySQL

ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass';

Debugging MySQL Connection Errors

- Start with broad grants (`'user'@'%'`)
- Refine from there

mysql -u myuser -p mypass -h myhost -P myport

```
jsnyder@db:~$ mysql -u myuser -h myhost
ERROR 1045 (28000): Access denied for user myuser@'10.10.10.10'
(using password: NO)

jsnyder@db:~$ mysql -u myuser -p -h myhost
Enter password:
ERROR 1045 (28000): Access denied for user
'myuser'@'10.10.10.10' (using password: YES)

jsnyder@db:~$ mysql -u myuser -p -h myhost
Enter password:
ERROR 2005 (HY000): Unknown MySQL server host 'myhost' (0)
```

Developing Against MySQL

We use:

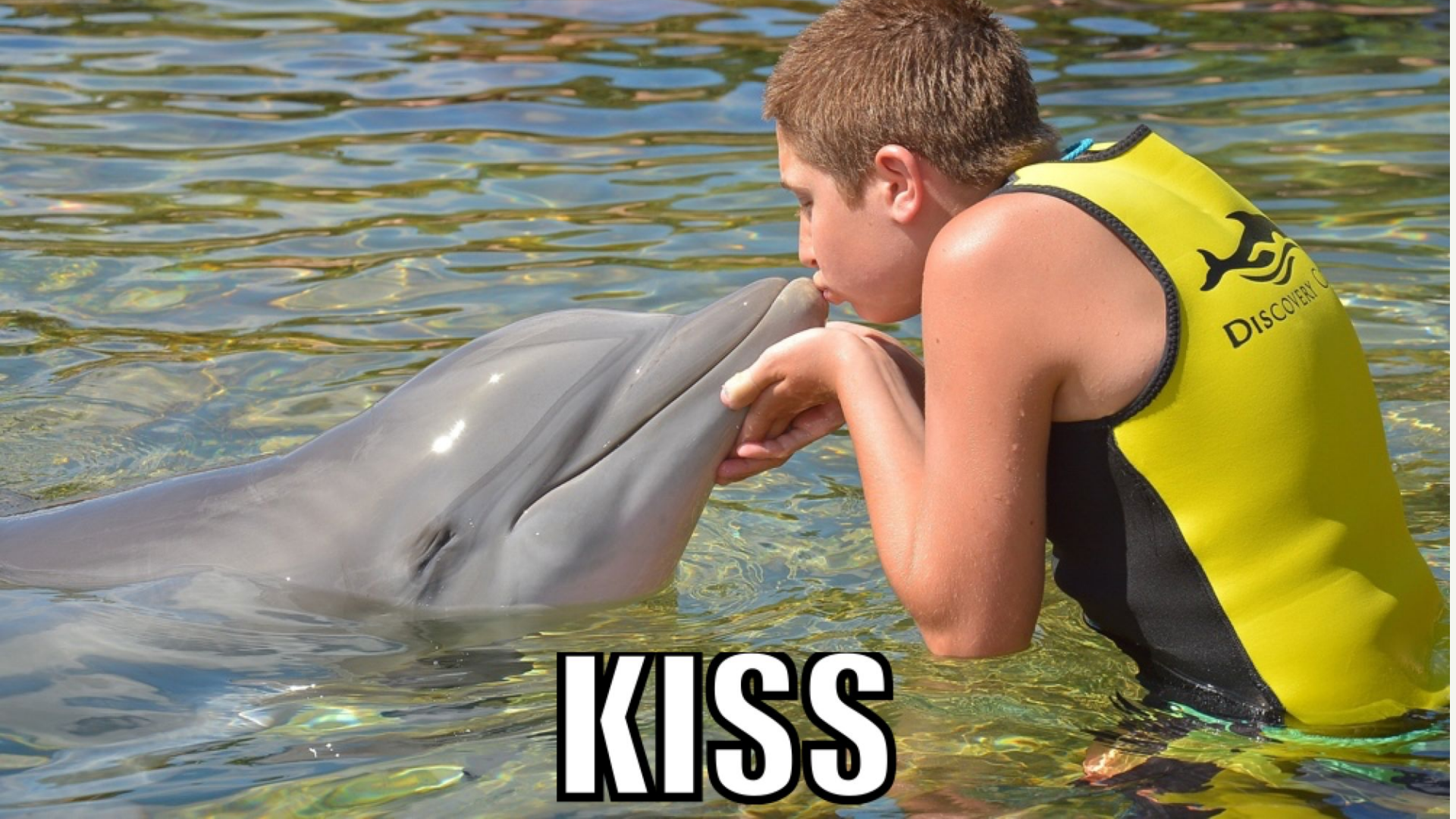
Language	Python
MySQL Client Library	mysqlclient-python (there is also Connector/Python)
Object Relational Mapping (ORM)	SQLAlchemy
Schema Manager	Hand-rolled system

To ORM or not to ORM?

About That Schema

- Have developers read up on normal form
  - It's OK to denormalize for performance (later)
- Use one table for one thing
- Try not to get too fancy right away

Reference: [Database Normalization Basics]


KISS

When Developing Against MySQL

- Don't swallow database client warnings/errors
- Track client error rates
- Make sure devs get error feedback

Build feedback to go directly to developers.

Application Connections

- You can start simple
- Reads and writes scale differently
- Use discovery, load balancer or proxy
  - Smartstack
  - ProxySQL
  - MySQL Router

Use technology you already have if possible.

Back to the Administration Part

- When you care about your database
  - Back it up to a safe place
  - Test restoring them
  - Have more than one

Different Kinds of Backups

- Binary a.k.a. hot backups
- Logical backups
- Point in time recovery
  - tl;dr: backup your binary logs

Binary Backups

- "They're Hot"
- MySQL Enterprise
- xtrabackup
  - Streaming

Logical Backups

- Spoiler alert: it's SQL
- `mysqldump`
- Best run on an offline/not replicating DB
- Sometimes the only way to upgrade

Backing up the Binary Logs

- Logs of all changes
- Can be used to "roll" a DB forward in time
- More on these later...
- Copy & compress

Sample command to look at these:

```
mysqlbinlog --base64-output=DECODE-ROWS -v mysql-bin.000001
```

Testing Backups

- Backups are only as good as your ability to use them
- Test your restores

Point in Time Recovery With Binary Logs

- `mysqlbinlog` turns binary logs into SQL
- Use the coordinates you saved
- It's documented, but I'm not aware of a great tool

```
jsnyder@db:~$ cat
/path/to/backup/xtrabackup_binlog_info
mysql-bin.000003      589767


jsnyder@db:~$ mysqlbinlog
/path/to/binlogs/mysql-bin.000003
/path/to/binlogs/mysql-bin.000004 \
    --start-position=589767 \
--stop-datetime="2017-10-29 09:00:00" | mysql
<args>
```

Practice!

- Try it before you need it
- We call these "war games"
  - And have failover exercises
- The best plan is one you know will work!

More on replication next.

But First, a Word About Words

MySQL uses the terms *MASTER* and *SLAVE*.

It is modern and inclusive to use *PRIMARY* and *REPLICA*.

This talk uses MASTER and REPLICA.

Replication

- Basics
  - One master + multiple replicas
  - Client-initiated, client configured
  - The replication user also needs grants

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE
  ON *.* TO 'replication'@'10.%' ...
```

Logs & Threads

- Master executes statement, writes it to:
  - Binary log
- Replica's IO_THREAD reads from master, writes it to:
  - Relay log
- Replica's SQL_THREAD executes statement, writes it to:
  - Binary log

Replicas can have their own replicas!

Basic Configuration

- Unique `server-id` is a *must*
- Binary log formats
  - `STATEMENT`
  - `ROW`
  - `MIXED`
- `Log-slave-updates = ON`
- `master_info_repository = TABLE`

MySQL won't execute events from it's own server-id.

```
jsnyder@db:~$ fgrep bin /etc/my.cnf
# binary logging is required for replication
log-bin                         = ../master_log/mysql-bin
relay-log                       = ../relay_log/relay-bin
binlog_format                   = statement


jsnyder@db:~$ fgrep server-id /etc/my.cnf
server-id                       = 171739394

jsnyder@db:~$ fgrep slave /etc/my.cnf
log_slow_slave_statements       = 1
log-slave-updates               = 1
slave_net_timeout               = 30
slave-skip-errors               = 1141
```

Configuring Replication

- `CHANGE MASTER TO`
  - `MASTER_USER =`
  - `, MASTER_PASSWORD =`
  - `, MASTER_HOST  =`
- If using GTID:
  - `, MASTER_AUTO_POSITION = 1`
- If not:
  - `, MASTER_LOG_FILE =`
  - `, MASTER_LOG_POSITION =`

Start it up

- `START SLAVE`
- `SHOW SLAVE STATUS`

```
mysql> SHOW SLAVE STATUS \G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.10.10.44
                  Master_User: replication_user
                  Master_Port: 3306
                Master_Log_File: mysql-bin.000008
            Read_Master_Log_Pos: 914832470
                 Relay_Log_File: relay-bin.000014
                  Relay_Log_Pos: 914832633
          Relay_Master_Log_File: mysql-bin.000008
               Slave_IO_Running: Yes
              Slave_SQL_Running: Yes
...
```

```
mysql> SHOW SLAVE STATUS \G
… (continued)
                   Last_Errno: 0
                   Last_Error:
...
          Exec_Master_Log_Pos: 914832470
...
        Seconds_Behind_Master: 0
...
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
```

What Could Possibly go Wrong?

- Replication errors in two places
- err.log

```
2017-10-28 17:22:06 5061 [ERROR] Slave SQL: Error 'Duplicate entry
'14' for key 'PRIMARY'' on query. Default database: 'yelp_db'.
Query: 'INSERT INTO user (id, name) VALUES ('14', 'Barbara')',
Error_code: 1062
```

- SHOW SLAVE STATUS:

```
   Last_SQL_Errno: 1062
   Last_SQL_Error:'Duplicate entry '14' for key 'PRIMARY'' on query.
Default database: 'yelp_db'. Query: 'INSERT INTO user (id, name)
VALUES ('14', 'Barbara')'
```

Try Breaking Errors Up

- Who had the problem?
  `[ERROR] Slave SQL: Error`
- What was it trying to do?
  `Query: 'INSERT INTO user (id, name) VALUES ('14', 'Barbara')'`
- What happened?
  `'Duplicate entry '14' for key 'PRIMARY'' on query.`
- Where was this?
  `Default database: 'yelp_db`

Common Causes of Errors

- Wrong binary log coordinates
- Inconsistent data

Monitoring - Server

- If you care about it, monitor it
- Lots of open source options
  - Built-in to your monitoring solution
  - percona-toolkit
  - Cheap scripts
  - Look on github

Humans like ~~pictures~~ graphs!

Host Stuff

- Disk space:
  - We file Jira ticket at 85%
  - And page at 95%
- Is MySQL running, does it seem OK?
- Are it's cron jobs completing?
- Does it's config match running state?

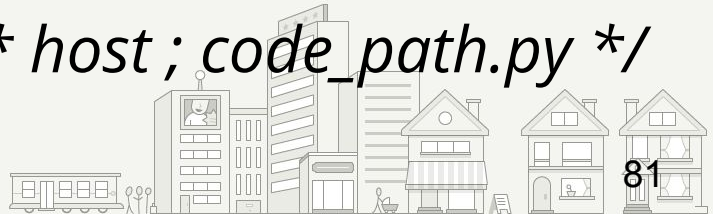
MySQL Server Stuff

- Auto-increment id limits
- Binary log growth
- Killed queries
- Replication hierarchy
- read_only
- Temporary tables on disk
- Threads connected
- Threads running
- More fancy replication checks

MySQL Client Stuff

- Have your application:
  - Track client errors
  - Track query times
  - Bonus: query annotations

select id, name from user where id = 1 / host ; code_path.py */*



Automated Defense

Kill and Log:

- Long-running queries
- Long-running transactions
- Too many of the same queries

Send feedback directly to developers via the client.

Handling Client Errors

- MySQL client will receive errors
  - And retry
  - Or fail *gracefully*
- Track client errors & statistics
- Alert devs where reasonable

Let developers see immediate feedback of a change!

Automated Testing

Before devs push to production they can test for:

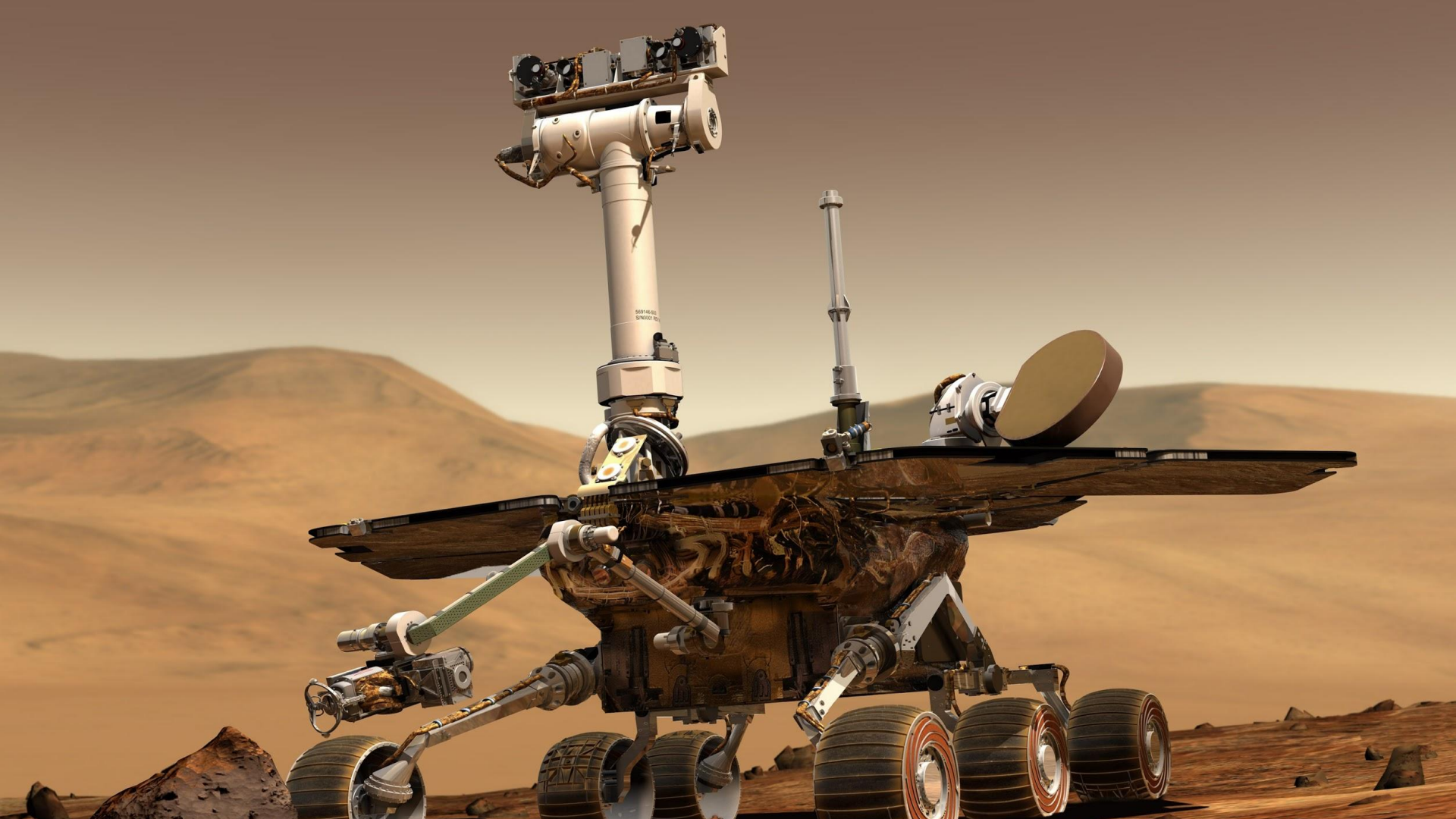
- Queries that will be slow
- Transactions that do too much

Send feedback directly to developers via the client.

Is All This Really Worth it?

Seems like a lot of work!

Let's moment to reflect on humans and feelings.



Think About it - Ops Perspective

If you get paged:

- And it was because of someone else's change
- You might feel #negative_emotion
- Especially if it happens again...

Think About It - Dev Perspective

If someone tells you they got paged:

- You do something about it, you didn't know
- But if it happens again...
- You might feel #negative_emotion

Useful Automation/Monitoring

Will:

- Let developers discover issues & fix on their own
- Faster iteration while developing
- Confidence when pushing
- Keep the pager for when you're really needed

<3 <3 <3


Questions
Answers

Let's Talk About It

Questions, comments, quemments?

We covered:

1. MySQL History, Philosophy, Community
2. Installation, Basic Configuration, Launching
3. Using it, Backing it up
4. Replication basics, Monitoring
5. Database Defense & fast feedback to developers

Before we Break

Here is what we'll cover when we get back

- Running in Production
- Being Prepared
- Advanced Monitoring and Investigation
- Fantastic Errors and Where to Find Them
- When Things Are "Slow"
- Advanced Tuning
- Replication and Scaling Out
- Discussion

And we're back

Part 2a:

- Running in Production
- Being Prepared
- Advanced Monitoring and Investigation
- Fantastic Errors and Where to Find Them

And we're back

Part 2b:

- When Things Are "Slow"
- Advanced Tuning
- Replication and Scaling Out
- Discussion

Welcome to Production!

Some things to think about:

- Expectations
- SLOs - uptime and query time
- Caching where you can
- Degrading gracefully

Be prepared, or prepare to be surprised!

Because We Measure Uptime in 9's

Nothing is up 100%

- Get PM's and devs involved
- Put features behind toggles
- Maintenance mode(s)
- Read-only mode


TIME
80 PHOTO ICONS
OUT OF THE BOX

Preparation For the Worst

Embrace your fears:

- Think about what can go wrong
- Plan a response - write Runbooks!
- Practice

I don't always test in production...

It's Not All Bad

Sometimes you have to do this anyway:

- Upgrades
- New hardware
- New features mean new tables & queries

Try to use the same procedures every time.

What Could go Wrong?

- Host failures
- Network failures
- Provider failures
- Overwhelming usage

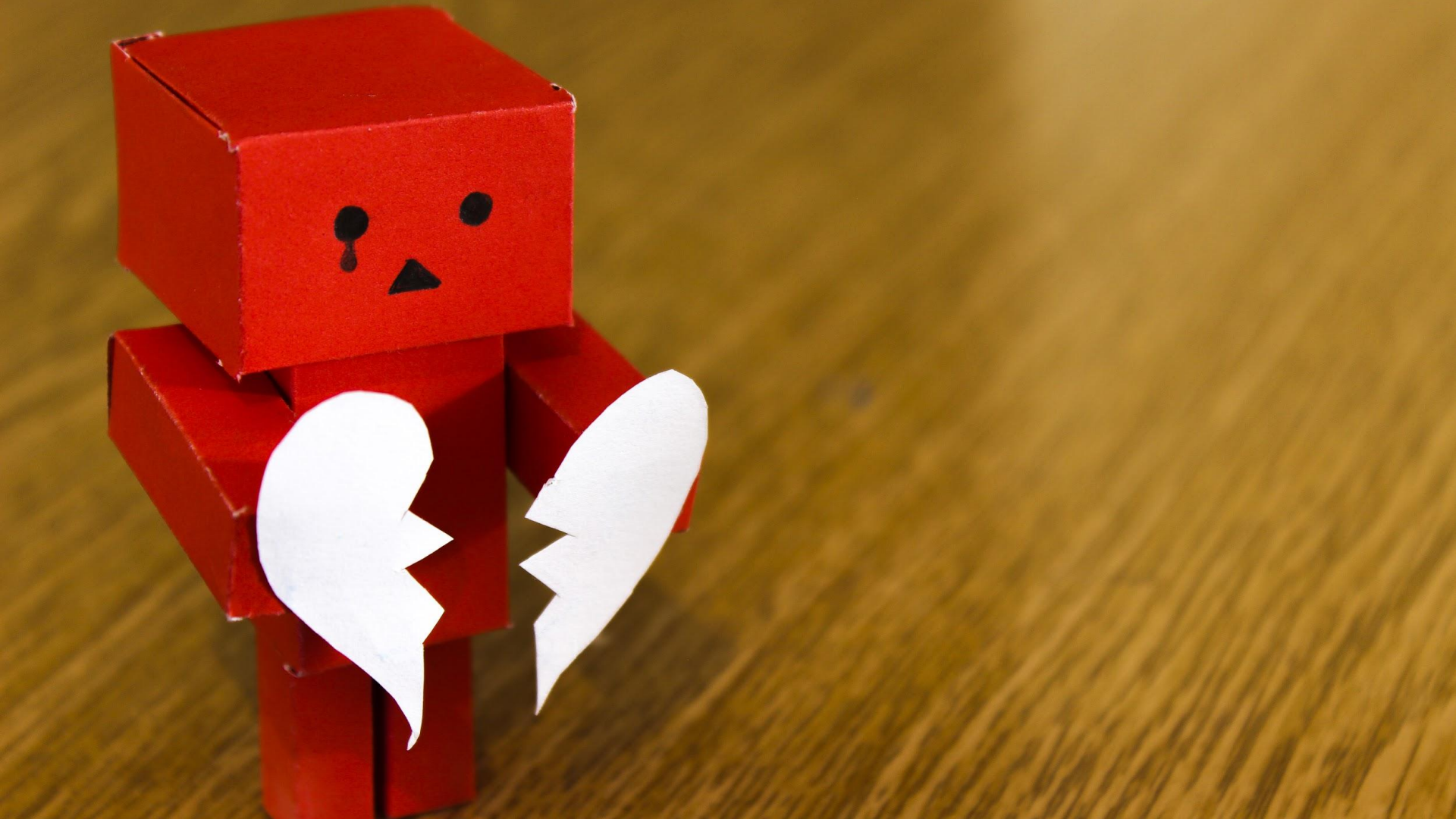
Best Practices - Server

Normalize maintenance & disaster recovery

- Replacing a replica
  - Same as building a new one
- Replacing a master
  - Same as promoting a new one
- Cloning a database
  - Same as restoring a backup

Best Practices - Application

- Give developers insight into DB health during:
  - Pushing code
  - Traffic surges



Learn From Mistakes

- Blameless Post-mortems
- Retrospectives


EUR/USD - 1,35379 - 00:00:00 14 giu (EEST)
EUR/USD (Bid), Ticks, # 300 / 300
23:35
23:40
23:46
Gold, spot - 1.276,820 - 23:00:00 13 giu (CEST)
Gold, spot (Bid), 1 minute, # 159 / 300, Logarithmic, Heikin Ashi
20:30 20:40 20:50 21:00 21:10 21:20 21:30 21:40 21:50 22:00 22:10 22:20 22:30 22:40
Alerts
Settings
Quote List [2]

Advanced Monitoring and Investigation

- Monitoring & alerting is pretty much
  - Collecting data
  - Using it

Monitoring the Server

Examples of things to monitor on a DB server:

- Is MySQL running?
- ~~Load~~ % CPU Idle
- Disk usage
- Critical processes
- Critical jobs

Monitoring MySQL

- Auto-increment id limits
- Binary log growth
- Killed queries
- Replication delay, hierarchy
- read_only
- Temporary tables on disk
- Threads connected
- Threads running
- Fancy replication checks

Monitoring MySQL Usage

SHOW GLOBAL STATUS has counters & gauges

- Queries per second
- Threads running
- Threads connected
- Slow queries
- Open transactions

Monitoring MySQL Client

You can monitor these through our client:

- Error rate
  - Rates per type
- Log all warnings & errors

https://www.flickr.com/photos/wocintechchat/25392405123/

Alerting

Ways your monitors can send you info:

- ~~Email~~
- Chat
- Tickets
- Pages

Chatbots are so hot right now.

Send Good Alerts

Make your Alerts Actionable

Make sure the alert:

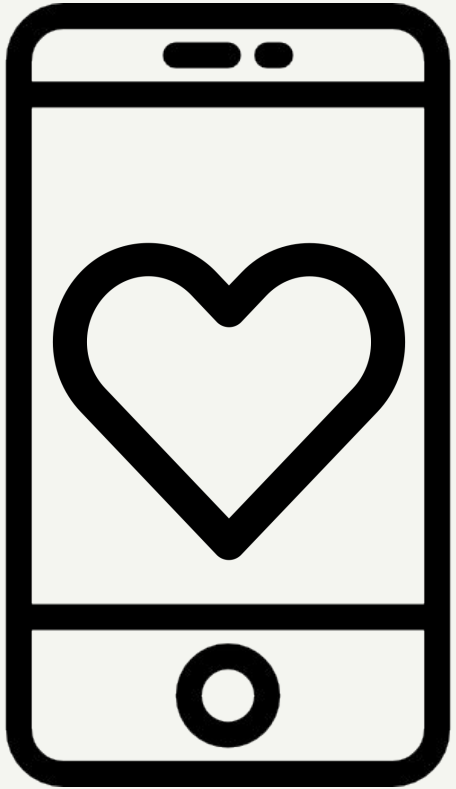
- Goes to the right team!
- Contains
  - A link to the runbook/guide
  - The first step to take

Iterate and Pay it Forward

Nothing is perfect the first time around

- Tune thresholds
- Keep track of alerts and response
- Handover to the next person down the line

If it happens consistently, automate the response!




ZZZ

Have a Runbook/Guide

- Start with the read some logs/part
- Iterate, refine, shorten

Tracking More Than Metrics

Cheapa$$ cron jobs capture this every minute:

- `ps`
- `mysql -e "SHOW PROCESSLIST"`
- `mysql -e "SHOW ENGINE INNODB STATUS"`

And this once a day:

- `mysql -e "SHOW TABLE STATUS"`
- `mysqldump --no-data`

What happened just before I was paged?

```
jsnyder@db:~$ cat ps-log.sh

#!/bin/bash

[ -d /var/log/ps ] || mkdir /var/log/ps

(
    date -R
    /bin/ps auxf
    free
    echo
) | bzip2 -c >> /var/log/ps/ps-`/bin/date +%Y%m%d-%H`.bz2
```

pt-stalk

Collect forensic data about MySQL when problems occur.

- Can run as a daemon
- Highly configurable
- Waits for a trigger condition
- Collects a ton of information

Fantastic Errors...

- Connection Errors
- Client Errors
- Server Errors
- Replication Errors

2013, 'Lost connection to MySQL server during query'

And Where to Find Them

- Your application's log
- MySQL's `error_log`
- `SHOW SLAVE STATUS`
- `SHOW ENGINE INNODB STATUS`

Just <insert search engine here> for it...

```
Traceback (most recent call last):
  File "/blahblah/blah/blah.py", line 357, in run
    return self.execute()
  File "/blah/blah.py", line 359, in execute
    return action(**match)
  File "/blah/blaah.py", line 172, in blaah_html
    return self.xtmpl('blaah.blaah',
env=self.blaah_env(blaah_form_obj))
  File "/lib/blah-blah/orm/session.py", line 470, in __exit__
    self.rollback()
  File "/lib/blah-blah/bleh.py", line 238, in commit
    self._delegate.commit()
```

OperationalError: (2013, 'Lost connection to MySQL server during query') None None

Debugging Lost Connections

- Try running the query itself
  - Does it return a huge number of rows? Sometimes?
  - Return a large amount of data?
  - Increase `max_allowed_packet`
- Does it go away if you increase `net_read_timeout`?
- Is the application server overloaded?
- It *might* be the network

MySQL Goes Away...

- Sometimes MySQL goes away...

OperationalError: (2006, 'MySQL server has gone away')

When an Error Contains a Query

A sensitive subject:

- You may store sensitive information in your DB
  - Email address
  - First & last names
  - Payment data
- You probably don't want it in your logs
- `pt-fingerprint`

Server Errors

You'll find these in the error_log

- log_error_verbosity = 1
- start/stop messages
- replication information
- crashes

InnoDB: Warning: a long semaphore wait:

```
--Thread 140375092094720 has waited at row0purge.cc line 770
for 241.00 seconds the semaphore:
S-lock on RW-latch at 0x1346820 '&dict_operation_lock'
a writer (thread id 140375041206016) has reserved it in mode
exclusive
number of readers 0, waiters flag 1, lock_word: 0
Last time read locked in file row0purge.cc line 770
Last time write locked in file
/dist/work/percona-server/storage/innobase/row/row0mysql.cc
line 3827
InnoDB: ###### Starts InnoDB Monitor for 30 secs to print
diagnostic info:
InnoDB: Pending preads 5, pwrites 0
```

MySQL Crashes

- MySQL will dump what info it can to the log
- `mysqld_safe` will attempt to restart MySQL
- Crashes are not normal

16:22:00 UTC - mysqld got signal 11 ;

```
This could be because you hit a bug. It is also possible that
this binary or one of the libraries it was linked against blah
blahblah blah blah blah blah blah.

key_buffer_size=268435456
read_buffer_size=131072
max_used_connections=8
max_threads=41
thread_count=2
connection_count=2
It is possible that mysqld could use up to
key_buffer_size + (read_buffer_size +
sort_buffer_size)*max_threads = 309930 K  bytes of memory
Hope that's ok; if not, decrease some variables in the
equation.
```

Common Crash Causes

- OOM Killer
- Corrupted Data
- Hardware Failure
- Bug

When Things Are "Slow"

Lots of things can cause this:

- Individual statements can be slow
- Lots of queries
- Lots of slow-ish queries

Find out what resource is in contention.

Tuning an Individual Query or Table

Examples here are from our open dataset.

Not queries we use in real-life :)

```
# Time: 2017-10-27T19:38:53.671918Z
# User@Host: root[root] @ localhost []  Id:    13
# Query_time: 4.947666  Lock_time: 0.000416 Rows_sent: 5
Rows_examined: 610939
SET timestamp=1509305933;
select count(*), r.stars
  from review r
  join business b on r.business_id = b.id
  join category c on b.id = c.business_id
 where c.category = 'Soul Food'
 group by r.stars ;
```

```
mysql> select count(*), r.stars from review r join
business b on r.business_id = b.id join category c on
b.id = c.business_id where c.category = 'Soul Food'
group by r.stars ;
+----------+-------+
| count(*) | stars |
+----------+-------+
|     2052 |     1 |
|     1785 |     2 |
|     2356 |     3 |
|     5432 |     4 |
|     8779 |     5 |
+----------+-------+
5 rows in set (4.95 sec)
```

```
mysql> EXPLAIN select count(*), r.stars from review r join
business b on r.business_id = b.id join /*snip*/ ... \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: category
   partitions: NULL
         type: ALL
possible_keys: fk_categories_business1_idx
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 587969
     filtered: 10.00
        Extra: Using where; Using temporary; Using
filesort
```

```
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: business
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 68
          ref: yelp_db.c.business_id
         rows: 1
     filtered: 100.00
        Extra: Using index
```

```
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: review
   partitions: NULL
         type: ref
possible_keys: fk_reviews_business1_idx
          key: fk_reviews_business1_idx
      key_len: 68
          ref: yelp_db.c.business_id
         rows: 25
     filtered: 100.00
        Extra: NULL
3 rows in set, 1 warning (0.00 sec)
```

```
mysql> pager fgrep rows
PAGER set to 'fgrep rows'

mysql> EXPLAIN select count(*), r.stars from review r join
business b on r.business_id = b.id join category c on b.id =
c.business_id where c.category = 'Soul Food' group by r.stars
\G
          rows: 587969
          rows: 1
          rows: 25
3 rows in set, 1 warning (0.00 sec)
```

```
mysql> nopager
PAGER set to stdout

mysql> select 587969 * 1 * 25 ;
+------------------+
| 587969 * 1 * 25 |
+------------------+
|         14699225 |
+------------------+
1 row in set (0.00 sec)


# From our slow query log, the numbers can be pretty off!
# Query_time: 4.947666  Lock_time: 0.000416 Rows_sent: 5
Rows_examined: 610939
```

```
mysql> alter table category add key (category) ;
Query OK, 0 rows affected (1.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table review drop key fk_reviews_business1_idx ,
add key( business_id, stars) ;
Query OK, 0 rows affected (38.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> alter table category add key (category) ;
Query OK, 0 rows affected (1.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: c
   partitions: NULL
         type: ref
possible_keys: fk_categories_business1_idx,category
          key: category
      key_len: 768
          ref: const
         rows: 235
     filtered: 100.00
        Extra: Using temporary; Using filesort
```

```
mysql>  select count(*), r.stars from review r join business b
on r.business_id = b.id join category c on b.id = c.business_id
where c.category = 'Soul Food' group by r.stars ;
+----------+-------+
| count(*) | stars |
+----------+-------+
|     2052 |     1 |
|     1785 |     2 |
|     2356 |     3 |
|     5432 |     4 |
|     8779 |     5 |
+----------+-------+
5 rows in set (0.03 sec)
```

Changing the Schema IRL

- Long ALTER TABLE's can hurt performance
- pt-online-schema-change
- gh-ost

And Just a Bit More on Indexes

There's a a great manual page *How MySQL Uses Indexes*

`PERFORMANCE_SCHEMA` can give you stats on index usage.

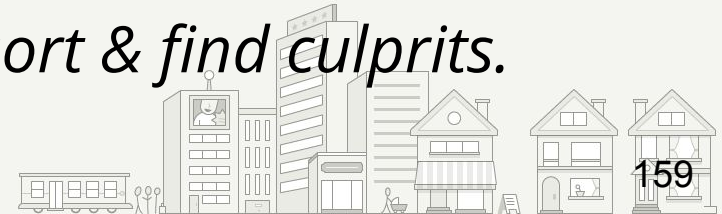
Keep in mind:

- *Adding indexes slows down writes a bit*
- *Cardinality*
- *Leftmost-prefixing*

Lots of Queries at Once

Find them using `SHOW PROCESSLIST`
Or SELECT … FROM `information_schema.processlist`

You can use cheap bash commands to sort & find culprits.



```
mysql -uroot -e "show full processlist" | awk '{print $5}' |
sort | uniq -c | sort -gr | head

jsnyder@db:~$ mysql -uroot -e "show full processlist" | awk
'{print $5}' | cut -c 1-70 | sort | uniq -c | sort -gr | head
    521 Sleep
    109 select count(*), r.stars from review r j
      3 Query
      2 Connect
      1 Command

# kill them with pt-kill. Use --print to test first!
pt-kill --config /etc/mysql/emergency-kill.conf --kill
--match-info 'your query' --daemonize --run-time=60
```

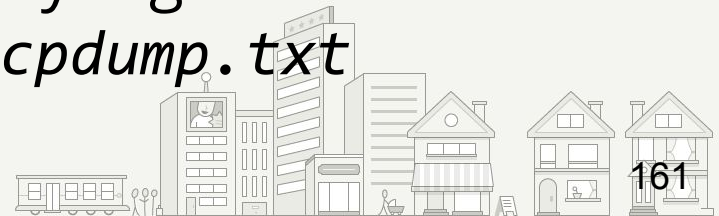
Finding the Next Thing To Optimize

You can:

- Some slow queries run infrequently
- Some slowish queries run all the time
- Visualization tools like Anemometer, VividCortex

Take a tcpdump & analyze with pt-query-digest:

pt-query-digest --type tcpdump tcpdump.txt




AFC
OFF
ON
TUNER INPUT
STEREO
FM 88 92 96 100 104 108 MHz
AM 530 600 700 800 1000 1200 1400 1600 kHz
TUNING
FM STEREO
M-AM TUNER
AUX
TUNER
PHONO
FUNCTION
MONITOR
MODE
HI-FILTER
LOUDNESS
SOURCE
TAPE
STEREO
MONO
OFF
ON
BASS
TREBLE
BALANCE
VOLUME

Tuning MySQL/InnoDB

- I/O
- Memory
- Tables/Transactions

I/O is Always the Slowest

InnoDB has a ton of knobs and dials

- Innodb_io_capacity
- Optimize your filesystem
- Add more memory, larger/more buffer pools
- Tinker with innodb_flush_method

There is a whole manual page dedicated to optimizing InnoDB disk I/O.

How InnoDB Uses Memory

Think of the InnoDB buffer pool as InnoDB's cache

- Increase the amount of memory to reduce I/O

Rough math:
innodb_buffer_pool_instances * innodb_log_buffer_size
+ (read_buffer_size + sort_buffer_size) * max_threads

MySQL Memory Calculator looks out of date, but can give you a pretty good idea...

On Transactions

To view running transactions:

- `SHOW ENGINE INNODB STATUS`
- `SELECT * FROM information_schema.INNODB_TRX \G`

```
mysql> SHOW ENGINE INNODB STATUS ;
...
------------
TRANSACTIONS
------------
Trx id counter 50806393604
Purge done for trx's n:o < 50806389839 undo n:o < 0 state:
running but idle
History list length 527
LIST OF TRANSACTIONS FOR EACH SESSION:

...
```

```
...

---TRANSACTION 50806392298, ACTIVE 12 sec starting index read
mysql tables in use 1, locked 1
314 lock struct(s), heap size 46632, 358 row lock(s), undo log
entries 353
MySQL thread id 23649033, OS thread handle 0x7f6bd31c7700,
query id 32707038971 10.10.10.40 application_user updating
UPDATE business SET ...
Trx read view will not see trx with id >= 50806392299, sees <
50806389830
```

Improving Transaction Efficiency

Reduce lock contention by:

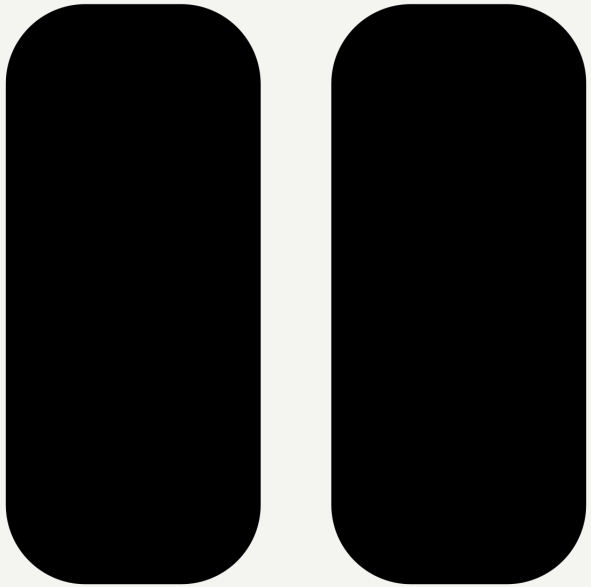
- Decreasing the number of locks your transactions use
- The total time of your transactions
- Making writes faster

See *InnoDB Locking Explained with Stick Figures*

Other Ways to Look at Transactions

- Performance_schema
  - `table_lock_waits_summary_by_table`

Let's Hit Pause For a Moment




https://www.flickr.com/photos/idhancock/8031897271

Replication and Scaling Out

Ways to use replication:

- Take load off the master
- Send reads to the replicas
- Having a separate reporting database
- Delayed replica with pt-slave-delay

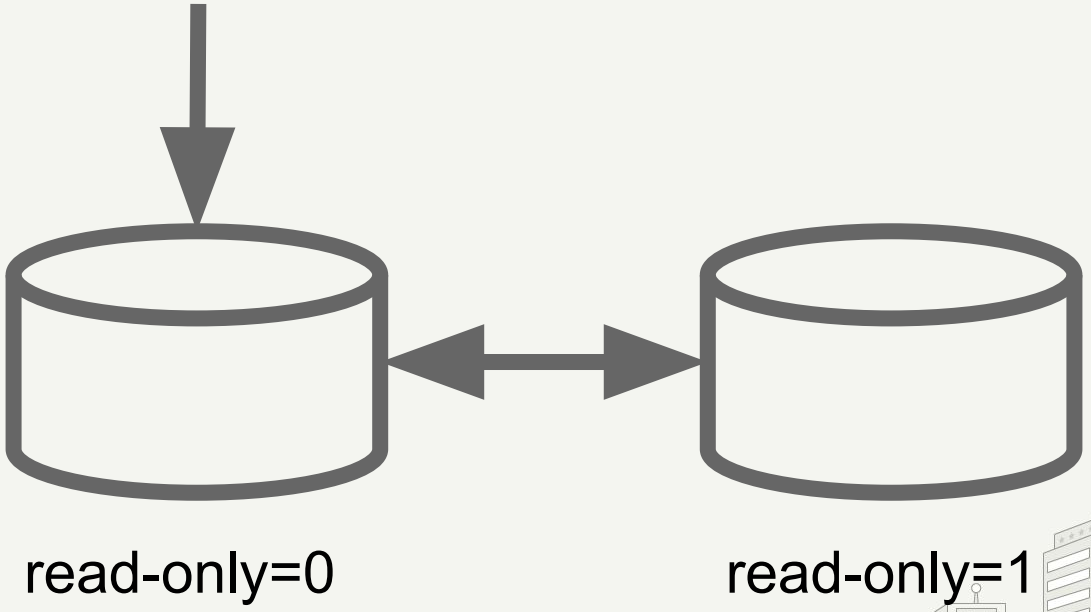
How MySQL Replication Works Again

- Master writes binary logs
- Each replica has 2 threads:
  - IO_THREAD: reads from master, writes relay logs
  - SQL_THREAD: runs SQL from relay logs
- Serial per database with parallel replication

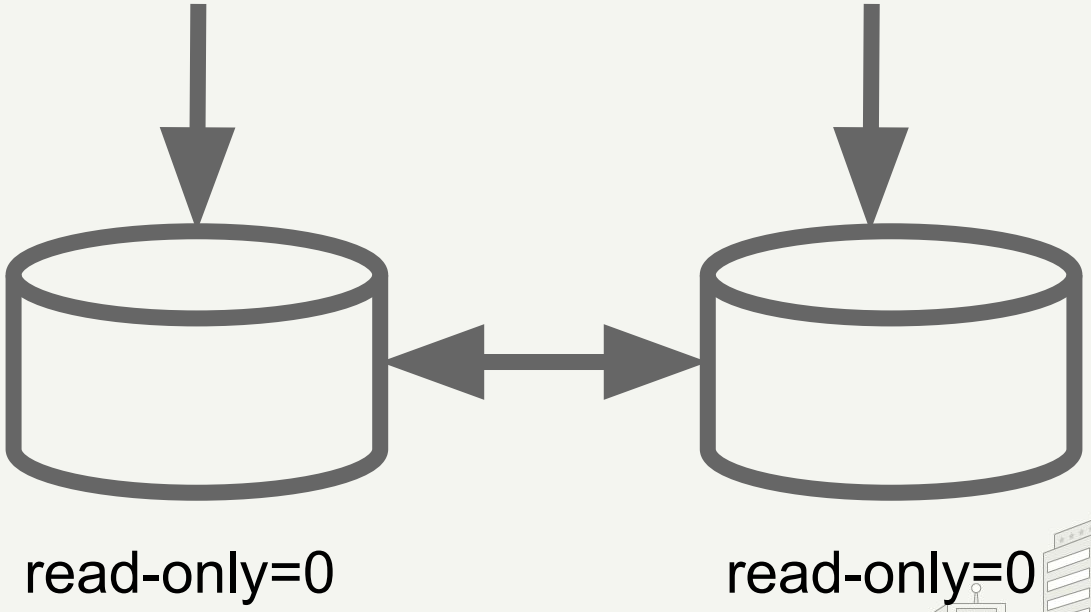
Some Replication Strategies

- master-master
  - One master "live", or
  - `auto_increment_increment`, `offset`
- master, intermediate master
- master(s) with many replicas

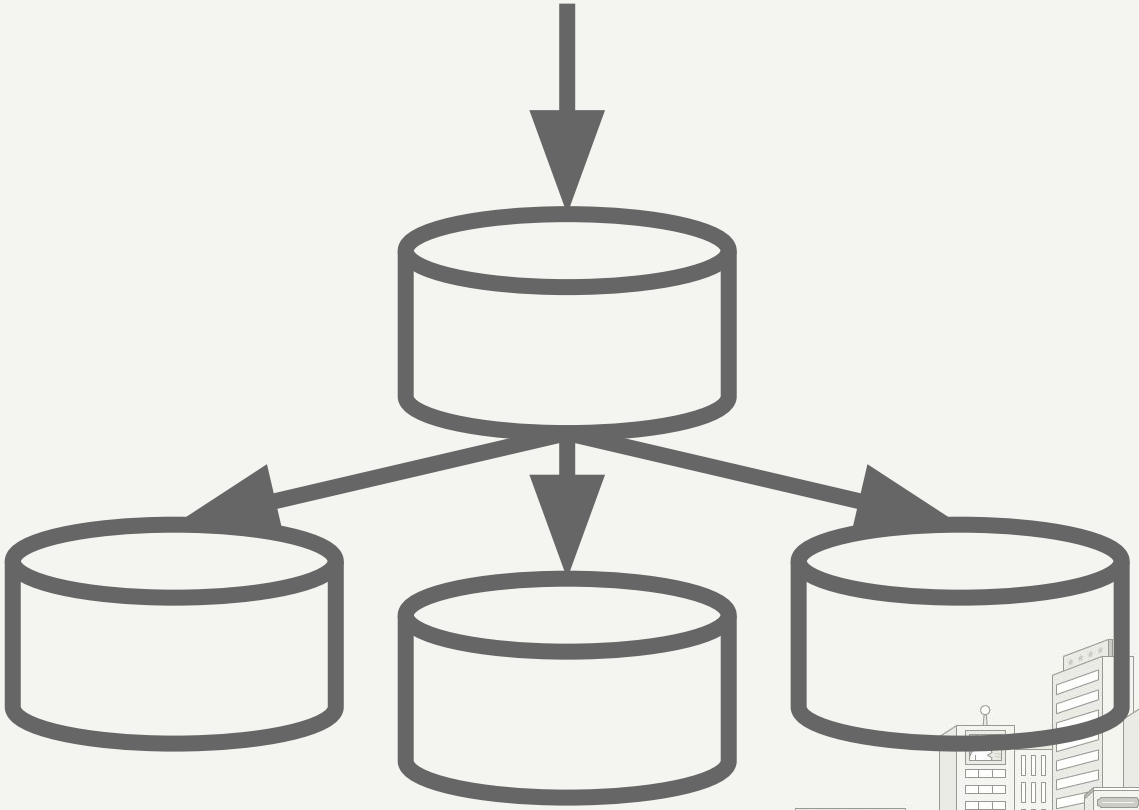
Master-Master - Active-Standby



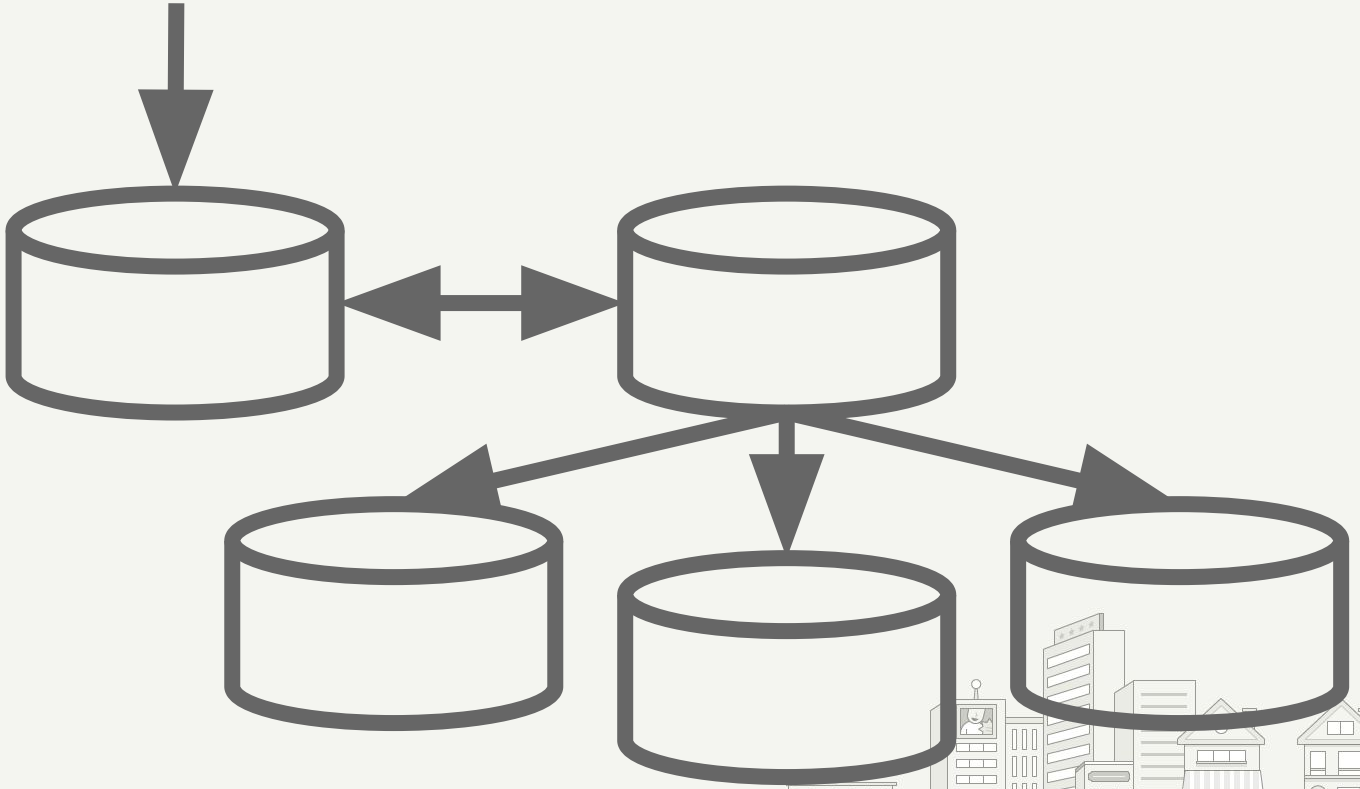
Master-Master - Active-Active



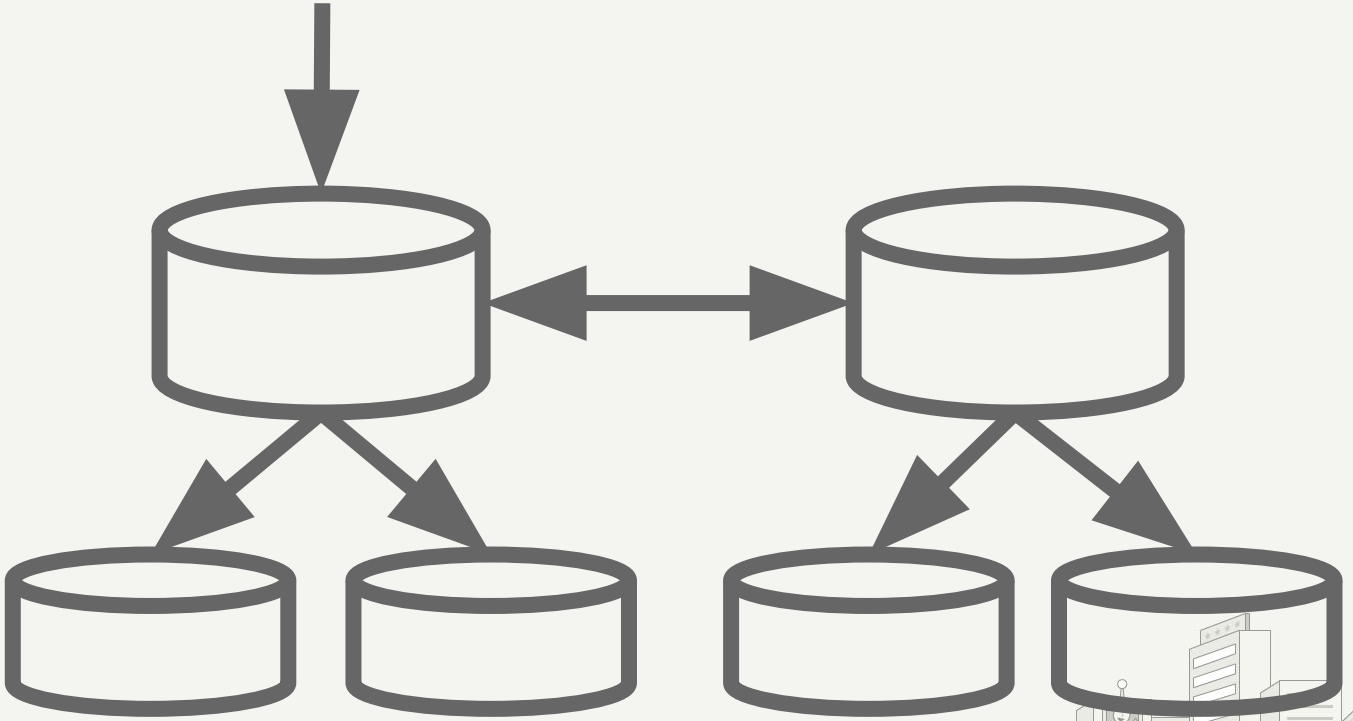
One Master, Many Replicas



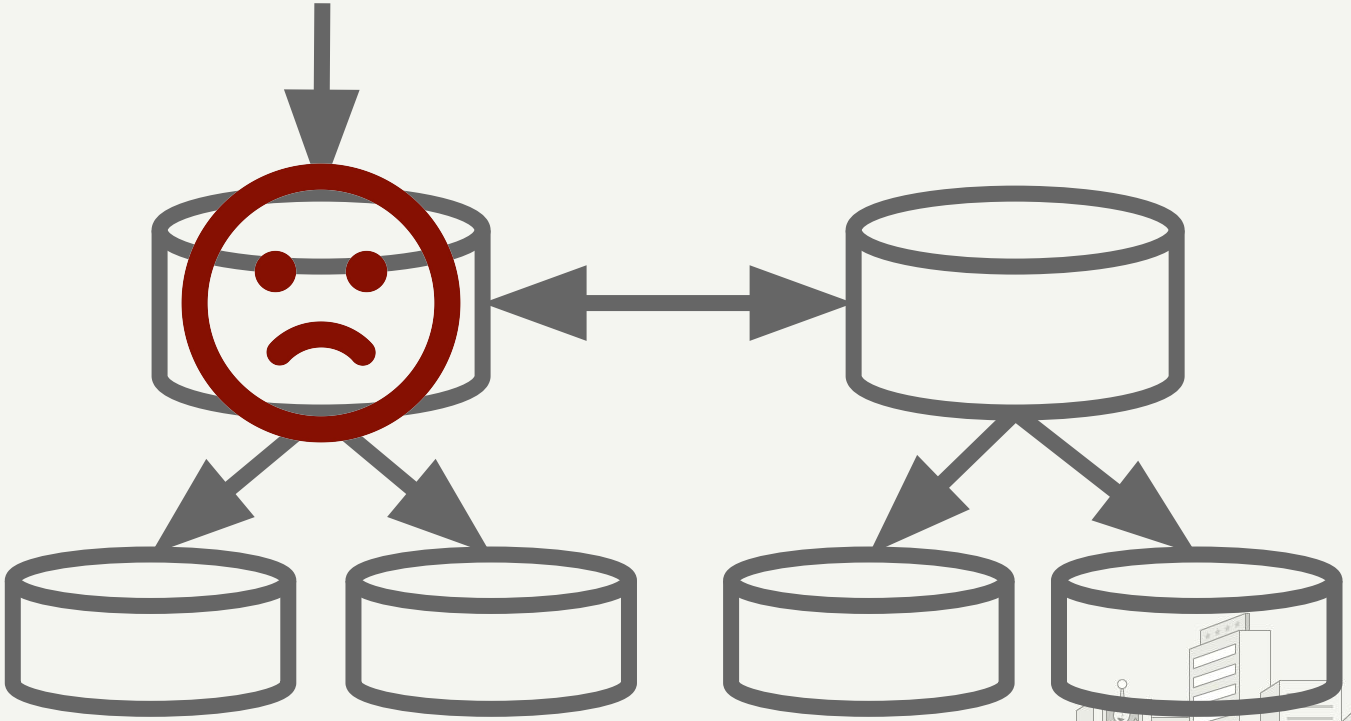
master, intermediate master



masters with their own replicas



Think About What You Want to Do



Tools For Replication

- Enable GTID - easy reparenting
- orchestrator - GUI and tooling
- pt-table-checksum - check consistency
- ProxySQL - can help you split your read-write traffic

When in Doubt, Throw it Out

- Don't write to sketchy databases
- Don't let your application write to replicas
- use `read_only=1` to prevent writes to replicas
- `SHOW MASTER STATUS`, `SHOW SLAVE STATUS`

Remember, a host that goes away may wander back...

Sample Master Promotion Plan

1. Set old master to `read_only=1`
2. `CHANGE MASTER TO` if necessary
3. Configure your application to write to new master
4. Set new master to `read_only=0`

Never allow writes to a replica unless you know that's what you want!


amazon.com

Speaking of Lag...

Common causes:

- IO_Thread can't keep up
  - Local load or slow/lossy network
- SQL_Thread can't keep up
  - Local load, long statements or transactions
  - `mysqlbinlog` to look at transactions

When You're Really Stuck

You can always phone a friend:

- MySQL has a big, friendly online community
- Oracle MySQL, Percona, MariaDB offer support

Get the most out of your interactions with experts by providing lots of context and what you've already tried.


Questions
Answers

Let's Talk About It

We covered:

- Running in Production
- Being Prepared
- Advanced Monitoring and Investigation
- Fantastic Errors and Where to Find Them
- When Things Are "Slow"
- Advanced Tuning
- Replication and Scaling Out


We're Hiring!
www.yelp.com/careers/

fb.com/YelpEngineers

@YelpEngineering

engineeringblog.yelp.com

github.com/yelp

Appendix - Links to Resources

- MySQL Manual
- MySQL Utilities
- Percona Toolkit
- Orchestrator: MySQL Replication Topology Manager
- ProxySQL: High Performance MySQL Proxy
- mysql-sys - tools for using performance_schema
- Anemometer - Slow Query Monitor

Appendix - Books

- *Database Reliability Engineering* by Laine Campbell, Charity Majors
- *High Performance MySQL* by Baron Schwartz, Peter Zaitsev & Vadim Tkachenko

Appendix - Blog Posts

- *Database Defense in Depth* by @geodbz
- *GrossQueryChecker: Don't get surprised by query performance in production! by @jcsuperstar*
- *InnoDB Locking Explained with Stick Figures* by @billkarwin
- *Blameless PostMortems and a Just Culture* by @allspaw
- Literally any Percona Blog Post by @svetsmirnova

Appendix - Image Sources

- Pexels - Images licensed under CC0
- Icons from The Noun Project
- #WOCInTechChat
- The Jopwell Collection

Appendix - Yelp Dataset

- Yelp Dataset

Other datasets:

- MySQL's Employees Sample Database
- Percona's list of Sample datasets