The Database Is Down, Now What?

Jeremy Tinley

jtinley@etsy.com
@techwolf359

Etsy 55 Washington St. #512 Brooklyn, NY 11222 April, 2015

About Etsy

2005
Founded

685
Employees

29M
Items listed

1.4M
Active sellers

19.8M
Active buyers

$1.93B
Gross merchandise sales in 2014

Headquartered in the
DUMBO
neighborhood of
Brooklyn, NY

With additional offices in
Berlin, Germany
Dublin, Ireland
Hudson, NY
London, United Kingdom
Melbourne, Australia
Paris, France
San Francisco, CA
Toronto, Canada

And people buying or selling from
Nearly every country in the world

Agenda

Evolution of MySQL Architectures

MySQL at Etsy

Etsy Problem Resolution

Evolution of MySQL Architectures

Master with Standby

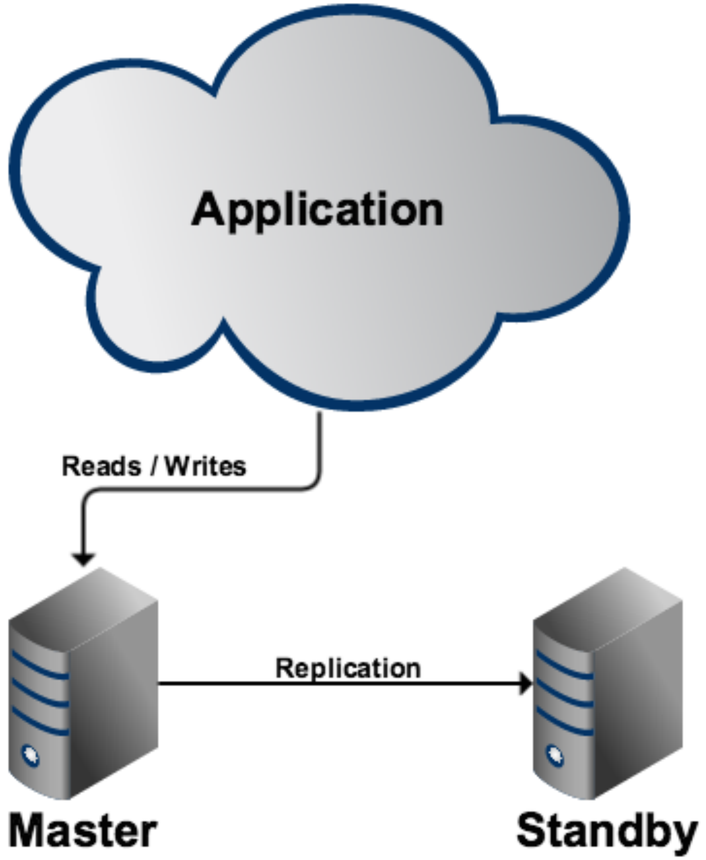
All reads and writes go to a single server.

Caveats:

- Doesn't scale. CPU, disk bottlenecks.
- Single point of failure.

What DBAs Do:

- Tune Queries (slow query log; pt-query-digest).
- Add caching (memcached; more memory for BP).
- Add replicas.

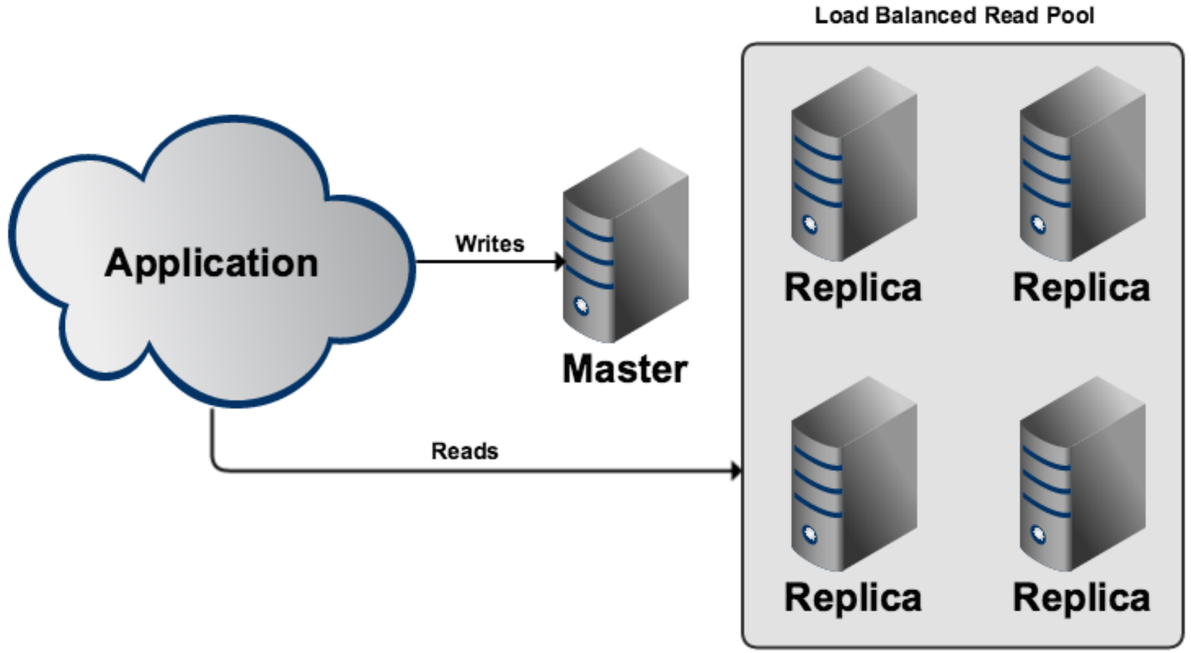


Master with Read Pool

All writes go to a single server. One or more read servers, likely using a load balancer.

Caveats:

- Doesn't address write bound workload.
- Replication lag becomes a problem.
- Single point of failure (still).



What DBAs Do:

- Shard.
- Use multi-threaded replication.

Functional Sharding

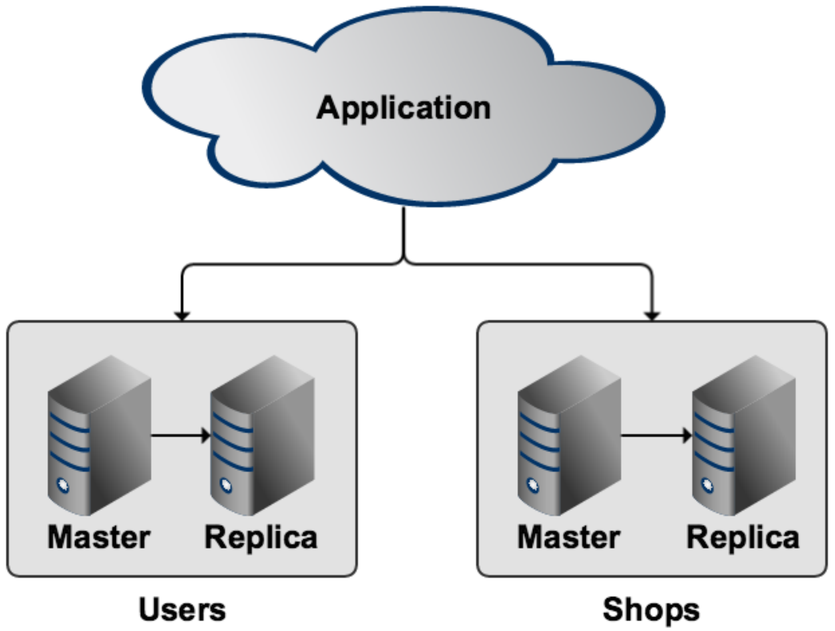
Take a heavy database or table and move the entire thing to another server.

Caveats:

- Scaling an issue when that object grows.
- Two different connections for some queries.
- Single point of failure (yes, still).

What DBAs Do:

- Logical Sharding.
- Buy bigger servers.



Logical Sharding

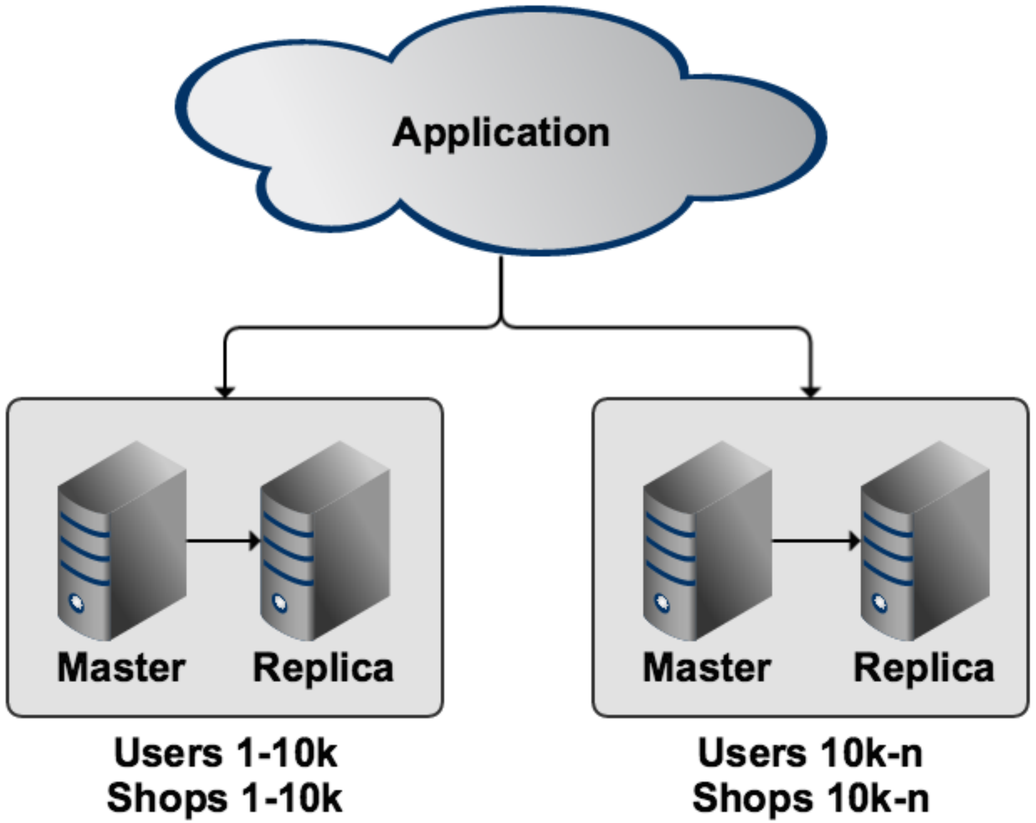
Multiple server pairs running the same schema, each with a subset of data in them. Can be done as range-based or mapping.

Caveats:

- Replication lag if using read pool.
- Mapping queries add a connection.
- Multiple single points of failure.

What DBAs Do:

- Master-Master, writing to both sides.



MySQL at Etsy

High Level Overview

http://ceilingcat.ninja

Etsy uses fairly standard MySQL patterns.

- CentOS 5/6/7
- Percona Server 5.5, no custom patches.
- Percona Server 5.6 in one environment for TokuDB
- Farm is managed by Chef
- Generally inspired by the Flickr design
- ORM (Not hand-crafted or vintage queries)

Tickets, Index & Shards

Our main data store consists of user-generated objects such as users, shops, listings, orders, favorites, etc.

- Ticket servers provides unique IDs for all objects
- Index servers provide the mapping for objects to shards
- Shard servers store the actual user data

Generating Globally Unique Object PKs

Ticket servers only purpose in life is to provide unique IDs for all objects we create in our environment.

- Pair of servers generating evens/odds using auto_increment_increment and auto_increment_offset. Code auto-retries the other side if unreachable.
- MyISAM table with an ID and a stub. Only 1 row for the latest ID.
- Use REPLACE INTO to get a new, unique ID for any object
- For more details: http://code.flickr.net/2010/02/08/ticket-servers-distributed-unique-primary-keys-on-the-cheap/

Index to Shard Locations

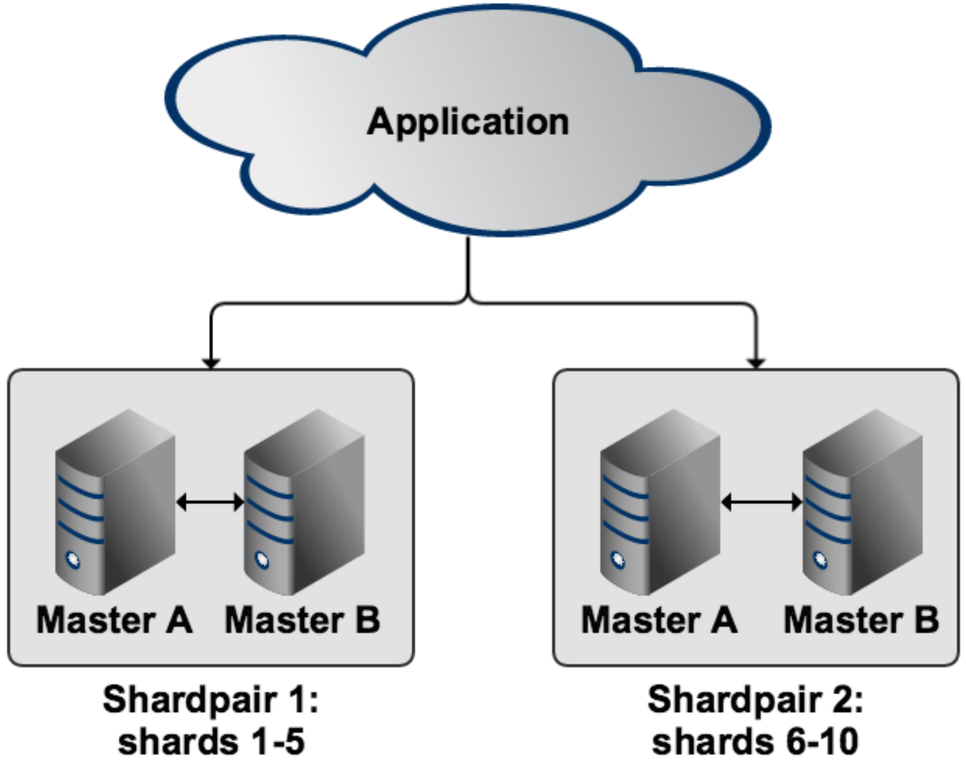
Once a new object is given a primary key, it's assigned a shard location.

- Index maps objects to shards.
- A general purpose memcache layer in front reduces connections/queries.
- Other convenient data stored here to prevent a subsequent shard lookup, such as user or shop name.
- Master-Master replication with writes to both sides. Incoming writes are use a simple mod 2 to ensure they pin to A or B side.

User Generated Data on Shards

The bulk of user data such as shops or listings lives in the shard farm.

- A shard pair is a pair (A/B) of physical servers
- Each server is running a single MySQL instance
- Each instance has multiple shard databases
- Replication is master-master
- Writes are done on both sides
- Shards are mod 2 driven, not range-based



Master-Master Isn't Scary.

In most environments, master-master replication is never a problem so long as you ensure you only write to one side. In our shard layout, we "pin" object reads/writes to one side and accept writes to both sides at the same time.

- Writes to an object will pin to A or B side and replicate to the opposite side
- Reads pin the same way making the application immune to normal replication lag
- Buffer pool working set is reduced to half of what it normally would be
- A single server going down impacts us half as much as normal

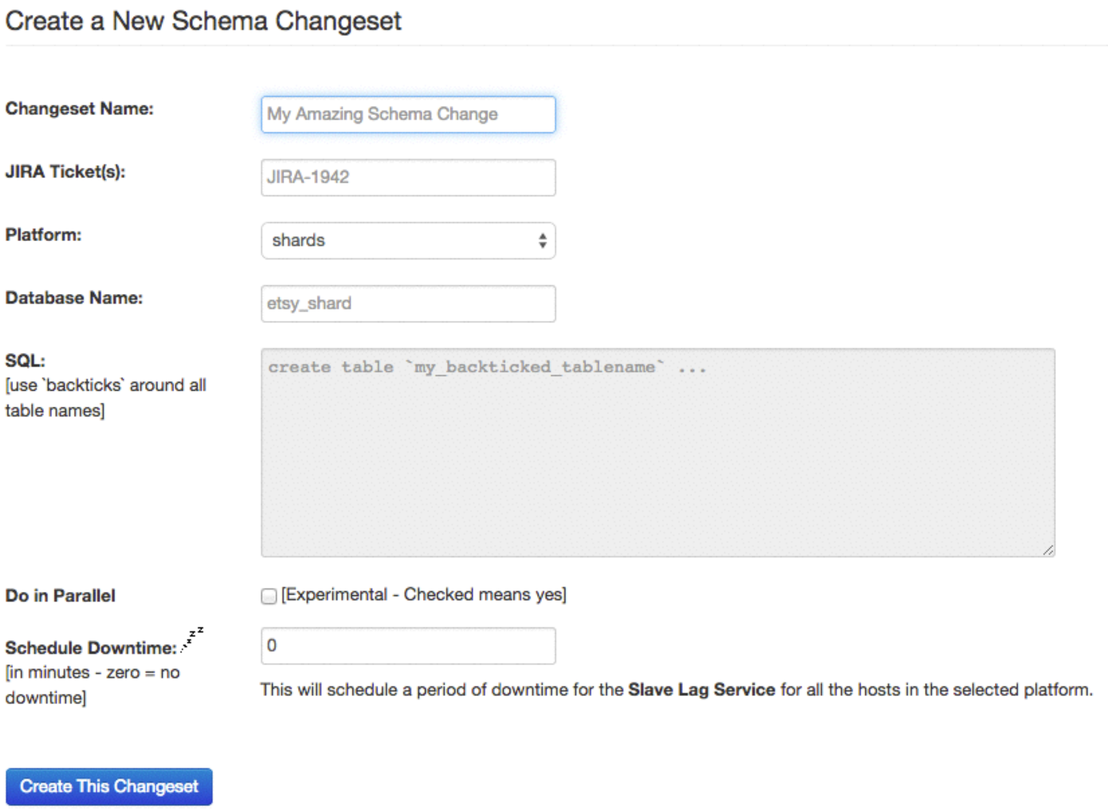
Maintenance – Schema Changes

We apply schema changes every Thursday. We use an internal tool called “Schemanator” to manage this process.

- Validates schema changes by applying to a temporary location
- Has validation for things like character sets, IDs as bigint
- Sets downtime for appropriate hosts/services
- Pulls side A, applies schema changes via Gearman jobs, reports back progress
- Repeat on B side, checksums when done
- Best part: code never forgets to SET SQL_LOG_BIN=0!

Schemanator Example.

Example of creating a new schema change.



Schemanator Examples

Changeset: sct-20150402-shard

This changeset was archived on Fri, 03 Apr 2015 00:58:43 +0000

WARNING: Some tables have **SQL Validation Errors**
See below for details.

WARNING: Some tables are **NOT listed in Shard Migrator Config**
See below for details.

Details of this schema changeset

Admin

Hash id	34abf393ad167c3f72e2a9124765638c
Changeset Name	sct-20150402-shard
Last Author	akachler
Deployer	akachler
JIRA	several
Creation Time	1427996286
Tables	[redacted]
# of Statements	11
Nagios Downtime	1200 minutes

Testing

Syntax Check	pass

Validation Errors (per table)

[redacted]
Table collation is not utf8mb4_unicode_ci
Potential _id column [redacted]s NOT a bigint unisgned

[redacted]
Table collation is not utf8mb4_unicode_ci
Table character set is not utf8mb4
Column `[redacted] is nullable
Column `[redacted] is nullable
Column `[redacted] is nullable
Column `[redacted] is nullable
Column [redacted]` is nullable
Column [redacted] is nullable
Column `[redacted] is nullable

Maintenance – Side Splitter

Schemanator has a tool inside of it called Side Splitter. This lets us pull a single server or an entire A/B side from production.

- Validation to ensure you aren’t pulling both sides of a pair
- Checks to see if you set monitoring downtime for the host
- Pushes a disabled connection file out to all hosts in parallel
- File is loaded within 1 second of landing on the host
- All new requests use the other side

Impact of Pulling a Server/Side

Pulling a server isn't without (brief) pain.

- Buffer pool for the new traffic is cold leading to an I/O spike
- Replication should be in sync, or very-nearly in sync. Failing to do so will serve stale data for the side of traffic that is being flipped over
- Used to create a single point of failure. If the remaining side went down during a schema change, it was messy to get restored.

The Need for More Copies

Realizing that having user data on just a pair of servers was scary, especially since we pull them once a week, we added more copies of data.

- Originally was a local delayed copy and an offsite copy in another data center
- This only got us a local, latent copy of the user data which had to be caught up before it was usable
- Natural progression was to store additional real time replicants, but server volume was prohibitive
- Consolidating lots of data onto less hardware while maintaining both replica sides was the goal

Enter Comboshards

The comboshards are standby servers that give us warm fuzzies when we pull half our masters every week.

- 1U Dell R630 960GB SSDs x24 across 2 controllers (18TB usable in RAID-6)
- Each physical server runs multiple instances of MySQL
- A side, B side and Delayed replicas divided up across the comboshard farm
- Primary fail-to source if we lost both A and B side masters from a shard pair
- Primary read location for non-user facing bulk tasks
- Bonus: They write ROW based binlogs which feed into Kafka for event triggering

R630 Form Factor

1U, 24x 960GB SSD

Future Me's Problems - Index

No architecture is ever perfect, or else we would all be doing it.

- Index race conditions. Needs code to fix it.
  - 2 shops created at the same time can fall on different sides of a shard and collide on a unique key.
  - Need globally unique tables to solve other hashing issues.
- Index scaling. We take a lot of connections per second.
  - Better/longer caching in front of Index, perhaps on web servers themselves
  - Functional partitioning of high traffic tables? Real read pools?

Future Me's Problems - Backup/Restore

We allow all developers access to production. This can be really scary because our time to restore feels too long.

- We use Percona Xtrabackup for hot backups alternating A and B every other night.
  - 75 minutes for network transfer (gigabit)
  - 20 minutes for —decompress —parallel=24
  - 60 minutes for —apply-logs —use-memory=128G
  - Replication catchup time varies. Usually hours.
- 5.6 + ROW + MyUndelete[1] can get us there faster, maybe?
- 10 gigabit network + pre-decompressed, pre-applied data directories?

1: https://github.com/lefred/MyUndelete

Our Biggest Problem – Hardware

Hardware will always fail. Often at 3AM.

- Pulling a server with Side Splitter is our failover process and it’s manual.
- Automated failover is scary. Quorum can be challenging.
- We limit damage by spreading our user data over a large number of servers. This, in turn, uses lots of power, rack space and time spent on management/automation.

A Walk Through Problem Resolution at Etsy

Step 1: Confirmation

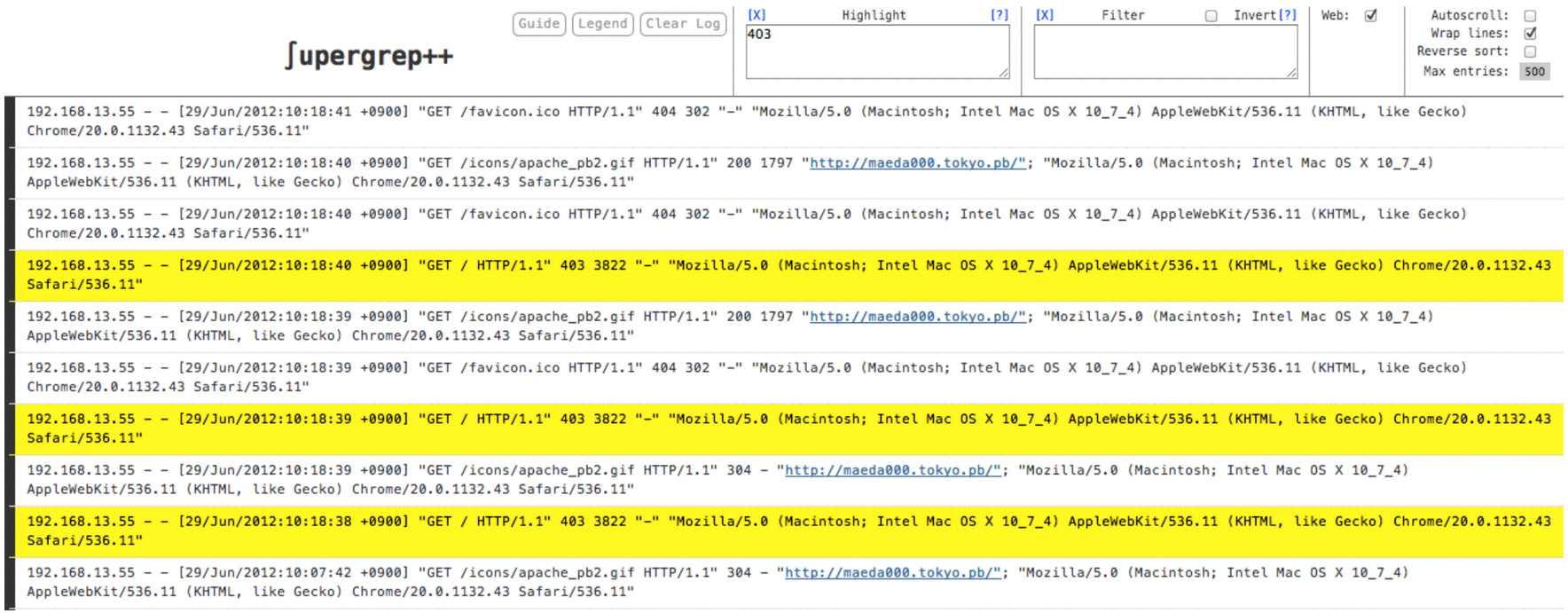
Is it Really a Problem?

Automated detection can fail.

- Can you reproduce it?
- Are you seeing the same errors?
- Is this an intentional change?
- What do the logs or graphs show?

Aggregate Log View with Supergrep

Supergrep is our filtering tool we use to view logs from multiple sources.



https://github.com/etsy/supergrep

Step 2: Tell Someone
(Even if its your fault)

“If you mess up, tell someone.
If you mess up big, tell everyone.”

The Importance of Communication

It takes virtually no time to communicate. The time you save from having to answer questions while you troubleshoot can be massive.

- 18 seconds to send an email
- 15 seconds to send an IRC message
- 16 seconds to send a text message

We have command line tools that let us post status updates that hit our status website (etsystatus.com) and twitter feed (@etsystatus) to push out important information to our members/users.

Step 3: Fix the Problem

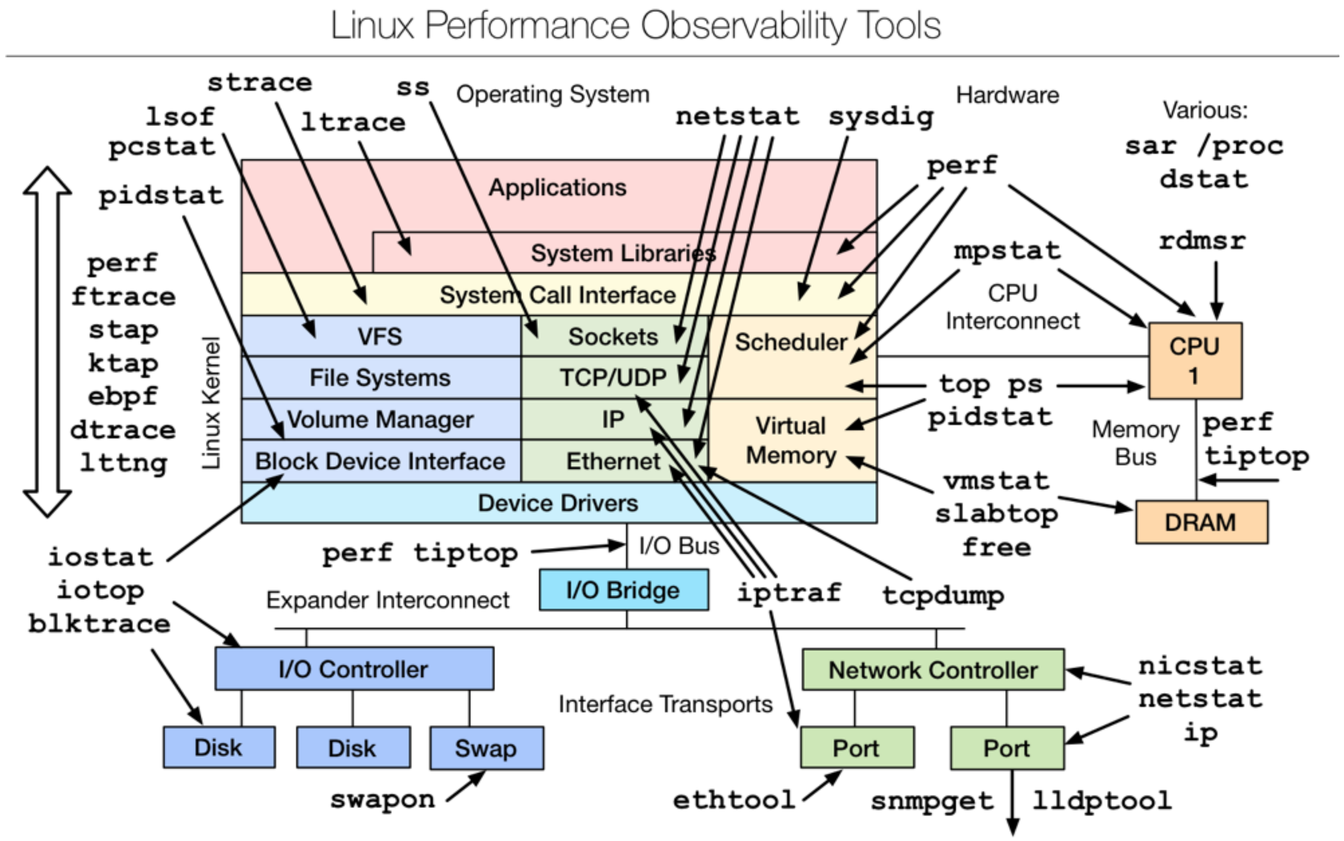
General to Specific

Start with the most general assumptions and work down from there.

- Hardware
- OS
- Software
- Look at hints provided from error messages.

Know Your Tools

Brendan Gregg's famous observability tools diagram.

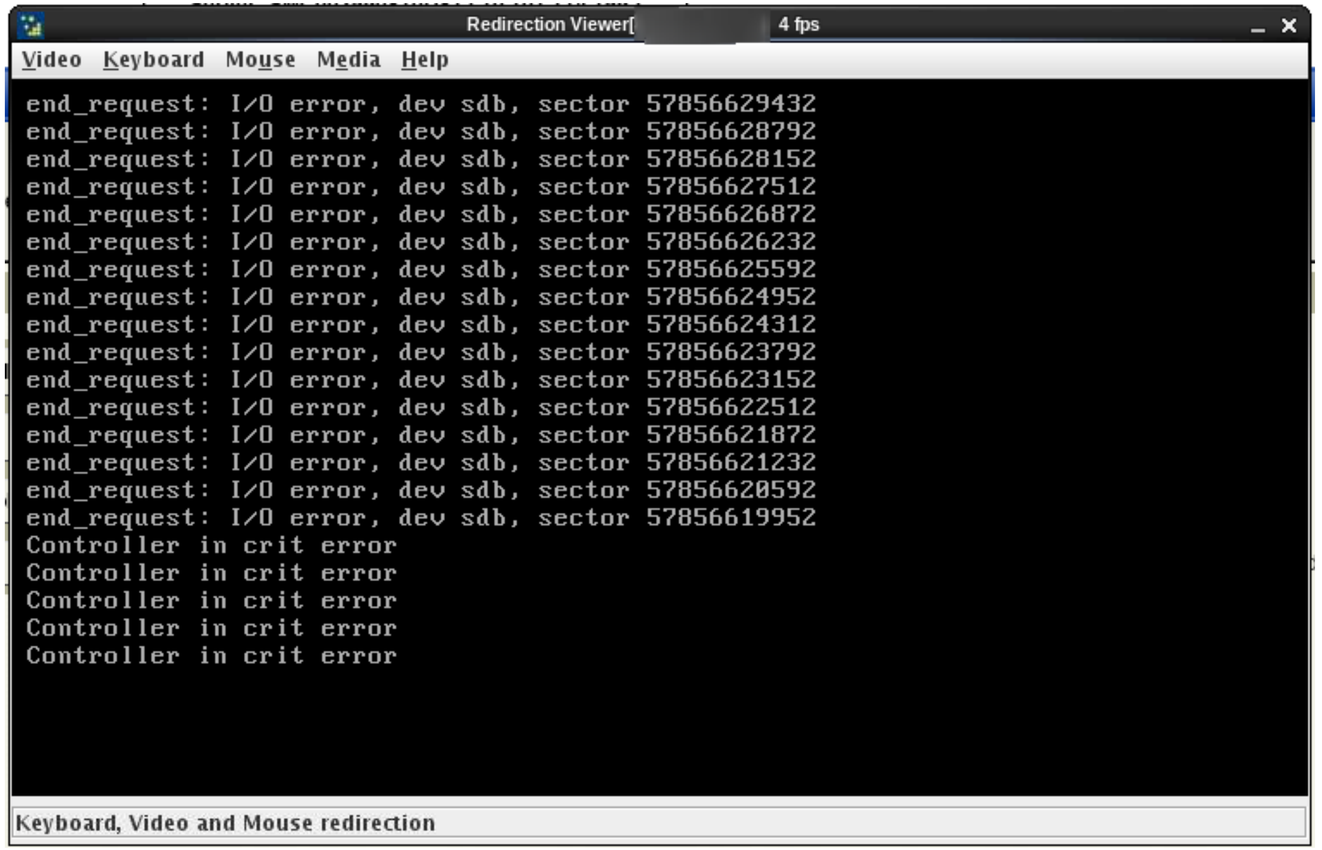


http://www.brendangregg.com/linuxperf.html 2014

Is Your Server Online?

Hardware failure happens. Detect and resolve.

- Is the server powered on & pingable?
- Can you SSH into the server?
- Check console or ILO/DRAC access.



Operating Systems – CPU Bound

From a Linux point of view, are there any resource constraints?

- top

```
top - 02:41:20 up 26 days,  8:16,  1 user,  load average: 2.98, 2.44, 2.39
Tasks: 518 total,   3 running, 515 sleeping,   0 stopped,   0 zombie
Cpu(s): 21.5%us,  2.6%sy,  0.0%ni, 72.1%id,  2.6%wa,  0.0%hi,  1.2%si,  0.0%st
Mem:  99025580k total, 98585760k used,   439820k free,     6508k buffers
Swap:  2047992k total,        0k used,  2047992k free,  8732916k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
64571 mysql     20   0 85.3g  76g 4972 S 508.3 80.7  89825:17 mysqld
```

- atop

```
ATOP - [redacted]                        2014/09/30  02:41:48                         ---------                          10s elapsed
PRC |  sys    4.07s |  user  32.49s |  #proc     514 |  #trun      8 |  #tslpi  1812 |  #tslpu     0 |  #zombie    0 |  clones     0 |  #exit      0 |
CPU |  sys      34% |  user    321% |  irq       14% |  idle   1969% |  wait     62% |  steal     0% |  guest     0% |  curf 2.29GHz |  curscal   ?% |
CPL |  avg1    3.30 |  avg5    2.57 |  avg15   2.44 |               |               |  csw   216582 |  intr  285487 |               |  numcpu    24 |
MEM |  tot    94.4G |  free  408.2M |  cache   8.3G |  dirty   5.0M |  buff    6.4M |  slab  718.6M |               |  vmcom  95.9G |  vmlim  49.2G |
SWP |  tot     2.0G |  free    2.0G |               |               |               |               |               |  vmcom  95.9G |  vmlim  49.2G |
DSK |          sda |  busy      60% |  read     1895 |  write    7750 |  KiB/r      7 |  KiB/w     29 |  MBr/s   1.38 |  MBw/s  22.24 |  avio 0.62 ms |
NET |  transport   |  tcpi  115488 |  tcpo  120848 |  udpi     367 |  udpo       7 |  tcpao     0 |  tcppo   3692 |  tcprs      2 |  udpip      0 |
NET |  network     |  ipi   115850 |  ipo   120850 |  ipfrw      0 |  deliv 115850 |               |               |  icmpi      0 |  icmpo      0 |
NET |  eth1    4% |  pcki    9436 |  pcko   40078 |  si  528 Kbps |  so   44 Mbps |  erri       0 |  erro       0 |  drpi       0 |  drpo       0 |
NET |  eth0    3% |  pcki  113376 |  pcko  109520 |  si   27 Mbps |  so   37 Mbps |  erri       0 |  erro       0 |  drpi       0 |  drpo       0 |

  PID     TID   RUID        EUID         THR    SYSCPU    USRCPU      VGROW    RGROW    RDDSK     WRDSK   ST   EXC   S   CPUNR    CPU    CMD          1/1
64571       -   mysql       mysql       1251     3.62s    22.59s         0K     240K   14184K    152.0M   --     -   S       0   263%    mysqld
```

- htop

```
  1  [|||||||||                 ]    7  [|||||                   ]    13 [||||                  ]    19 [||||||                ]
  2  [||||||||||||||||||        ]    8  [|||||                   ]    14 [||||||||||||||||||    ]    20 [||                    ]
  3  [||||||||                  ]    9  [||                      ]    15 [|||                   ]    21 [|||                   ]
  4  [|||||||||                 ]    10 [||                      ]    16 [|||                   ]    22 [||                    ]
  5  [||||||                    ]    11 [|||||||                 ]    17 [|||                   ]    23 [||                    ]
  6  [||||||||||||||||||        ]    12 [||                      ]    18 [||||||||||||||||||||  ]    24 [                      ]
  Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||87755/96704MB]   Tasks: 44, 1306 thr; 4 running
  Swp[                                                              ]   Load average:      2.51 2.42
                                                                        Uptime: 26 days, 08:16:56

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
64571 mysql      20   0 85.3G 76.2G  4976 S 403. 80.7    1497h /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
 6524            20   0 85.3G 76.2G  4976 S 18.0 80.7  0:26.96 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
64766            20   0 85.3G 76.2G  4976 S 13.0 80.7 45h52:49 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
42340 root       20   0  113M  3152  1288 R 11.0  0.0  0:01.40 htop
64767            20   0 85.3G 76.2G  4976 S  7.0 80.7 14h00:40 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
 6055            20   0 85.3G 76.2G  4976 S  7.0 80.7  2:41.52 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
51106            20   0 85.3G 76.2G  4976 S  7.0 80.7  1h21:11 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
 1123            20   0 85.3G 76.2G  4976 S  6.0 80.7  0:46.00 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/
```

Operating Systems - Disk/IO Limits

From a Linux point of view, are there any resource constraints?

- df - Are you out of space?
- iostat - Use -x to get Util%. Also look at svctime and await.
- sar (sysstat) - How do you compare historically?
- pt-diskstats - Shows you iops

Operating Systems - Memory Limits

From a Linux point of view, are there any resource constraints?

- free - Did you start swapping?
- vmstat - See swap bytes in/out
- NUMA - Non-Uniform Memory Architecture [1]

[1] http://blog.jcole.us/2010/09/28/mysql-swap-insanity-and-the-numa-architecture/

MySQL Specific

MySQL health can be tricky to find.

- Can you log into MySQL?
- Are you at max processes? Log in as a SUPER and SHOW PROCESSLIST
- Are there a long queries running?
- Locking? SHOW ENGINE INNODB STATUS
- Use tools to help consolidate information: innotop/mytop, VividCortex

Step 4: Talk About It

Learn From An Outage

Use this experience to improve the detection and resiliency in your environment.

- Document a Runbook
- Conduct a Postmortem
- Identify and Implement Remediation Items
- Game Days

Runbooks from Alerts

Document what people should do in case of an alert. Tie it in with your monitoring system.

- Brief explanation of how the alert works or what it checks
- Document how to determine if something is wrong
- Give examples of how to fix the problem
- Provide an escalation path; who do you contact if what you tried didn’t work

What is a Postmortem?

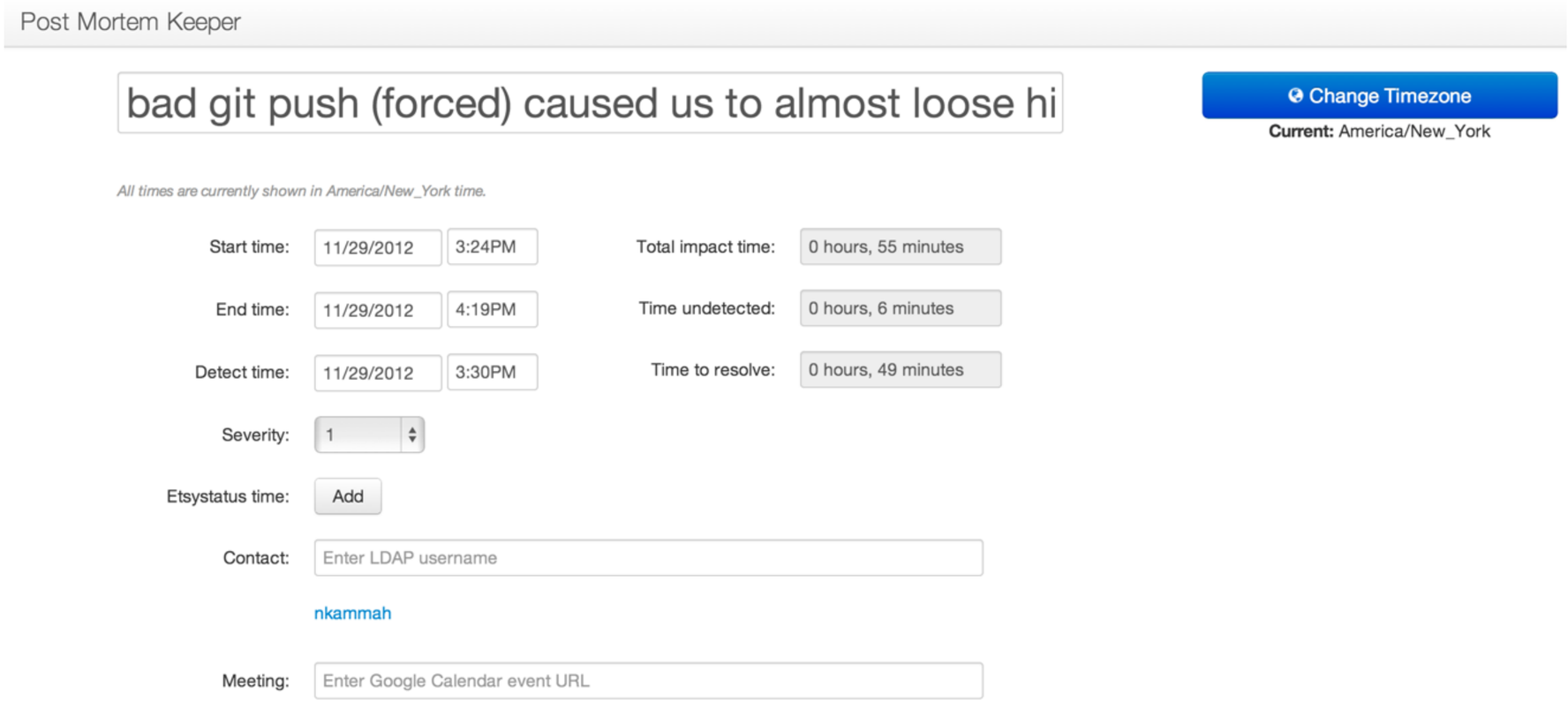
A review of the situation that resulted in the outage where teams can learn how to protect against repeat occurrences.

- Track it. Etsy open sourced morgue, our PM tracker [1]
- Keep it open. Invite as many people that want to attend.
- Keep it blameless. The point is to learn, not to point fingers.
- Keep it on topic. Don’t digress to what-ifs or should-haves.

[1] https://github.com/etsy/morgue

Morgue Example

Morgue tracks time, severity, timeline and several other elements associated with a PM.



Anatomy of a Postmortem

- Summarize What Happened
  - “The website went down because a configuration change was made to max_connections and we were unable to accept new connections.”
- Walk Through the Outage Timeline Step-by-Step
  - IRC Transcripts are Useful
  - Ask questions to determine why people came to (wrong) conclusions.
- Discuss Remediation Items
  - Track Them with a Ticketing System

Remediation

Make a complete list of the work necessary to inhibit future recurrence and set a deadline.

- Set a deadline. We do 30 days.
- Prioritize over everything else (within reason).
- Just Ship.

Game Days

There is no substitute for actual practice.

- What does your homepage look like if your database goes down?
- Fail over your database in a drill.
- Find the way your application will break if you do as many twisted things as possible to it.

Conclusion

Databases Go Down

DON’T hope it doesn’t happen. EXPECT it to happen. PLAN for what you do WHEN it happens.

- Know your architecture
- Know the pain points
- Document how to get services restored
- Review how to improve both your architecture and your processes

Questions?

Happy Sleeping

Etsy