pagerduty

# MAKING PAGERDUTY MORE RELIABLE USING PXC

PERCONA LIVE 2015

@dougbarth

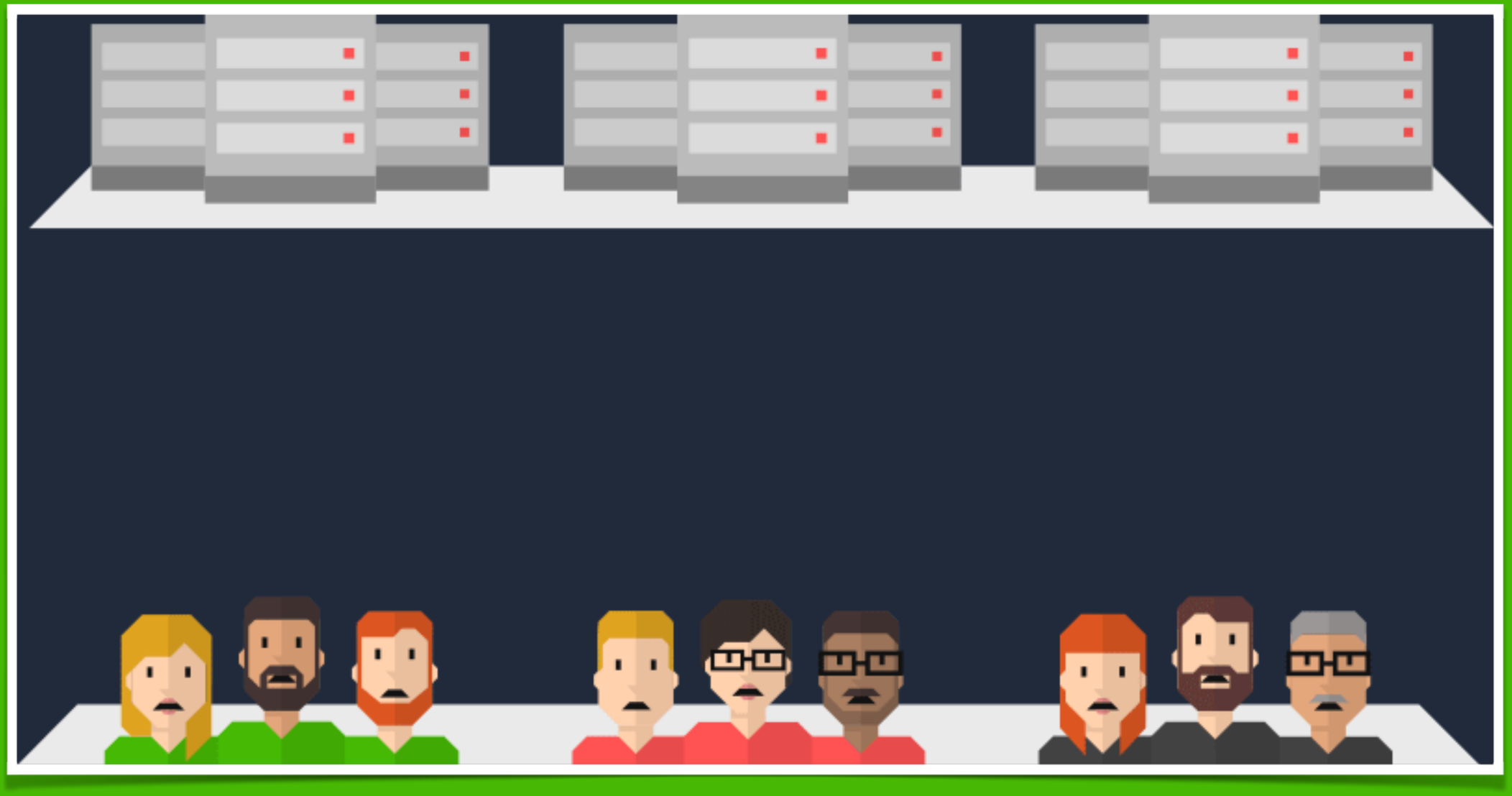
**Doug Barth**
@dougbarth

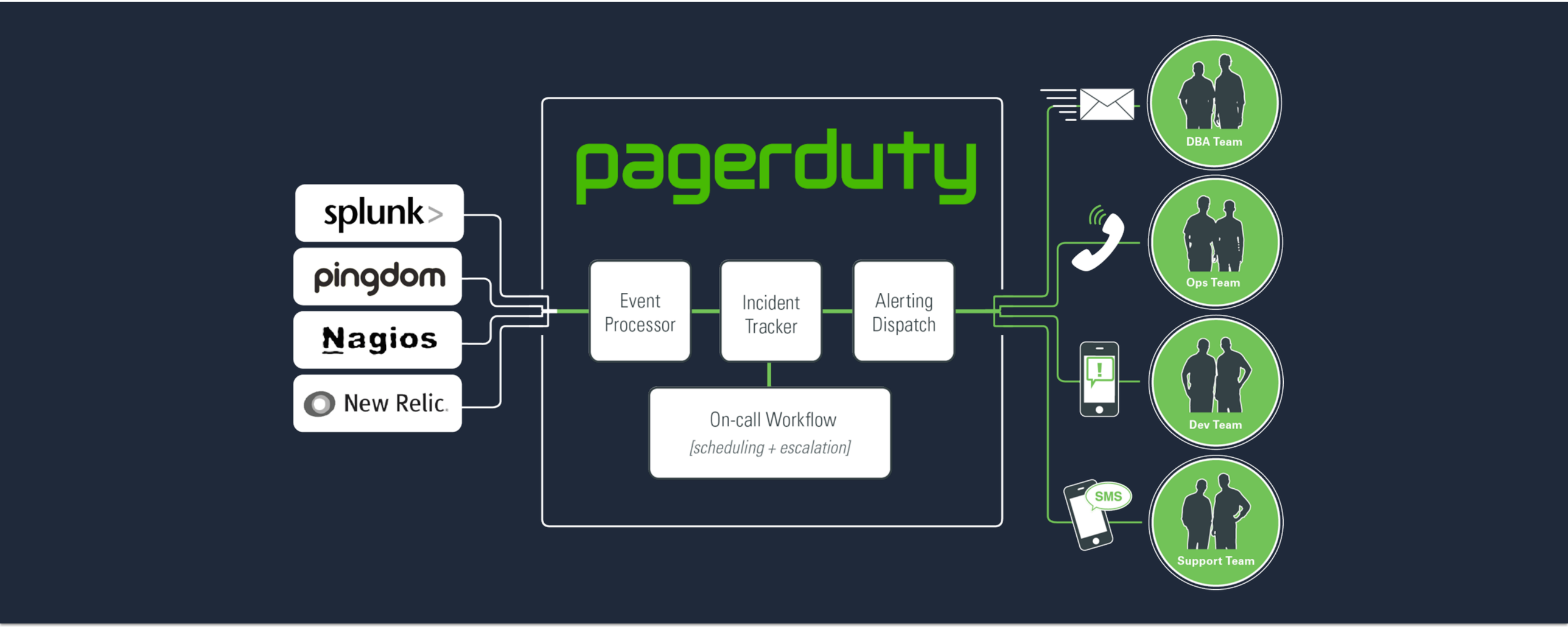
6yr: <finishes getting bow in hair>
3yr: Mama tied it like a shoe!
6yr: How do I look?
3yr: You look like a shoe.
6yr: <cries>

</scene>

FAVORITES
5

8:22 AM - 13 Apr 2015




splunk>
pingdom
Nagios
New Relic
pagerduty
Event Processor
Incident Tracker
Alerting Dispatch
On-call Workflow
[scheduling + escalation]
DBA Team
Ops Team
Dev Team
SMS
Support Team

# PagerDuty stack

| Then | Now |
| --- | --- |
| Monothilic Rails App | Rails & Scala |
| Cloud hosted | Cloud hosted |
| MySQL Community<br>(later Percona Server) | Percona XtraDB Cluster<br>Cassandra<br>Zookeeper |

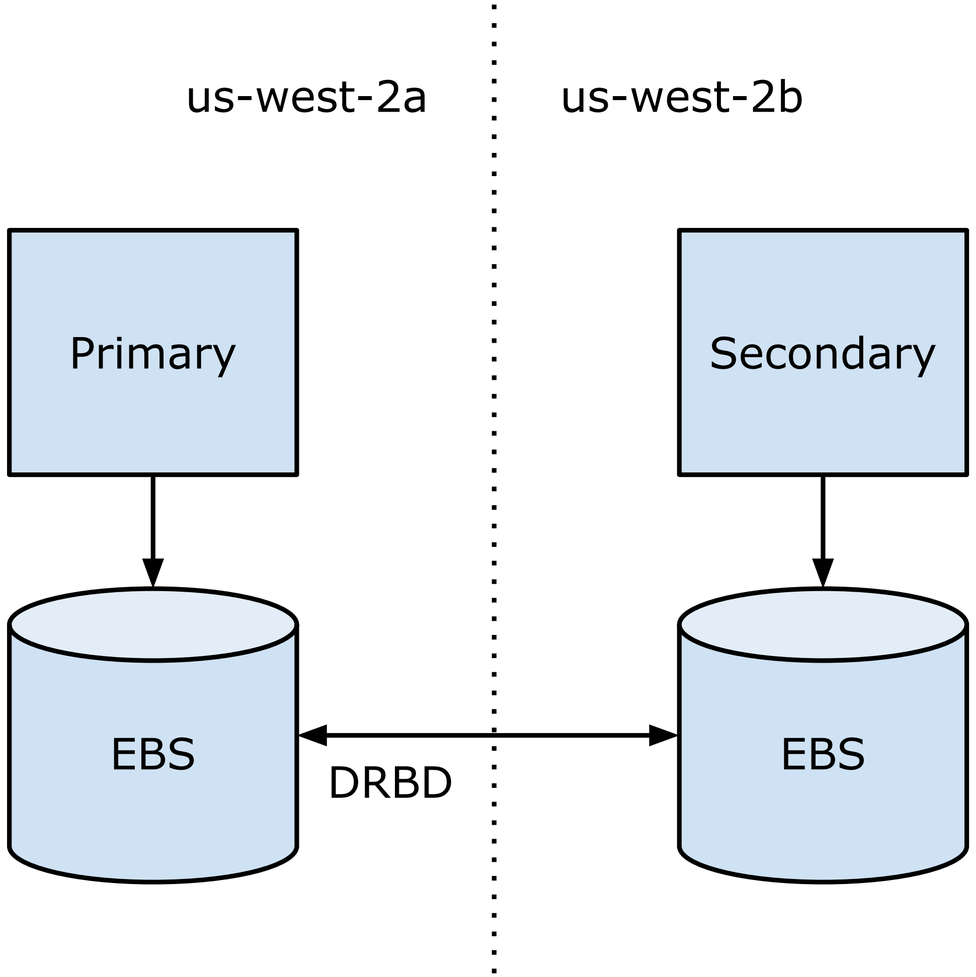
# MySQL at PagerDuty

| | |
|---|---|
| Data size | ~ 600 GB |
| Queries / s | 6,000 - 7,500 |
| Txns / s | 200 - 300 |

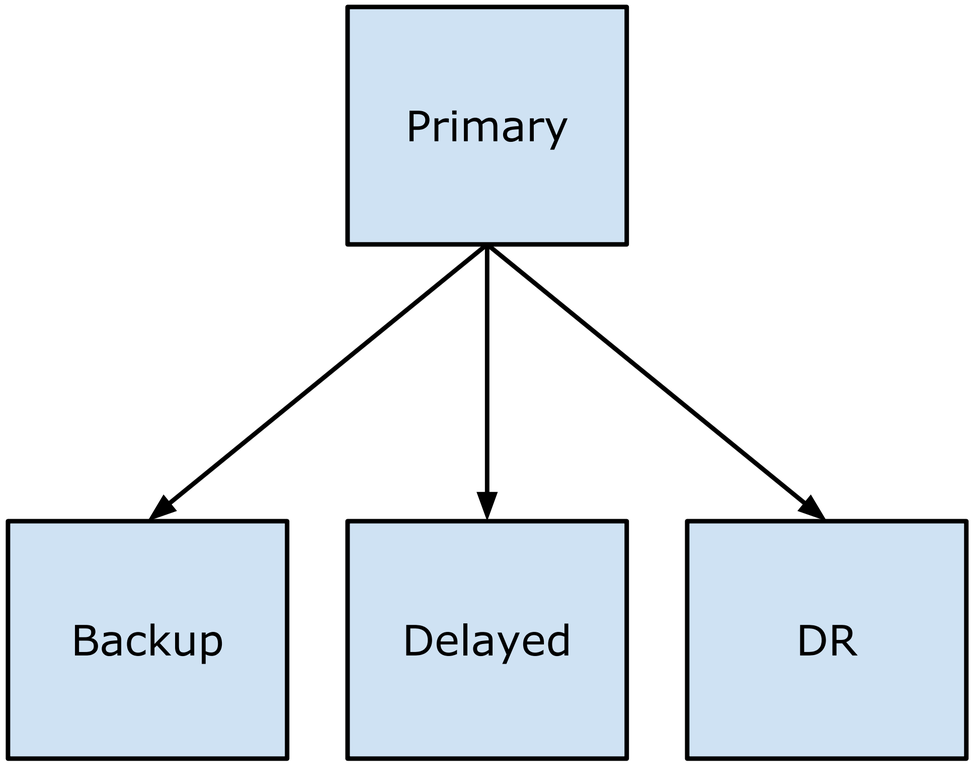
# Ye Olden Setup

***ca. Jan 2013***

# Ye Olden Setup



# Ye Olden Setup



# Problem #1

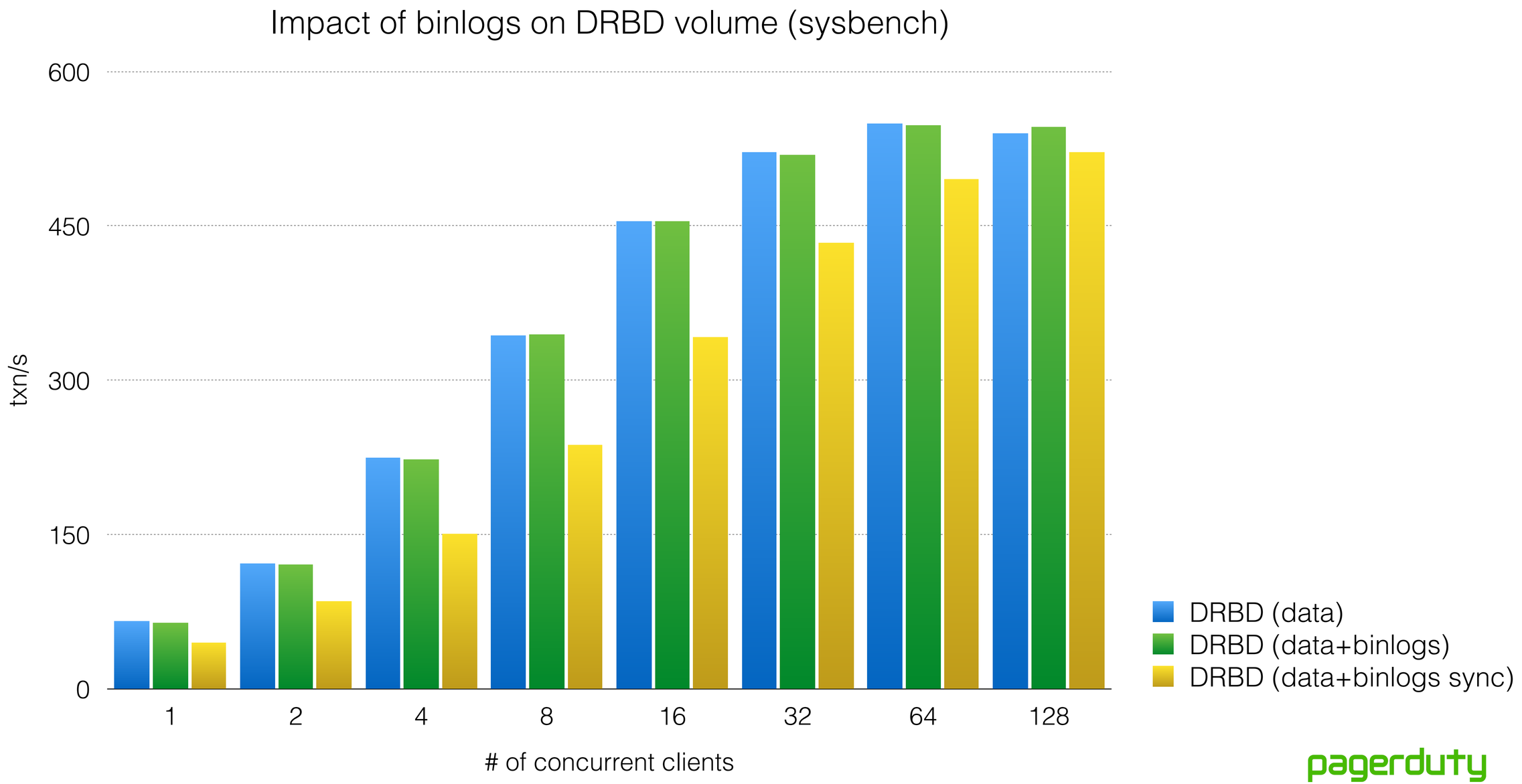
# DRBD Failover

1. FLUSH TABLES WITH READ LOCK
2. Stop MySQL on primary
3. Unmount DRBD volume
4. Set primary to secondary role
5. Flip EIP over to secondary
6. Confirm secondary is now primary
7. Mount DRBD volume on new primary
8. Start MySQL on new primary
9. Wait for clients to reconnect
10. Wait for buffer pool to warm up

# ~ 3 minutes of downtime

# Problem #2

# Broken replicas after flip


Impact of binlogs on DRBD volume (sysbench)
600
450
300
150
0
txn/s
1
2
4
8
16
32
64
128
# of concurrent clients
DRBD (data)
DRBD (data+binlogs)
DRBD (data+binlogs sync)

# Percona XtraDB Cluster

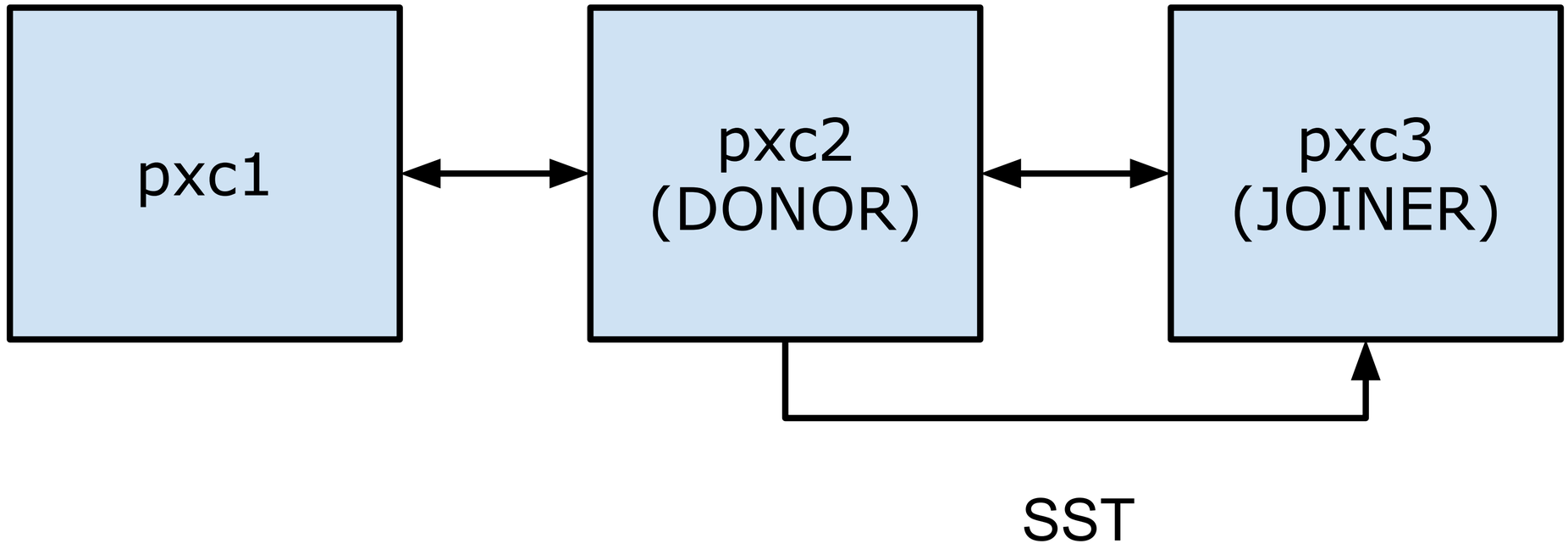
# Percona Server + Galera = PXC

# Multi-master

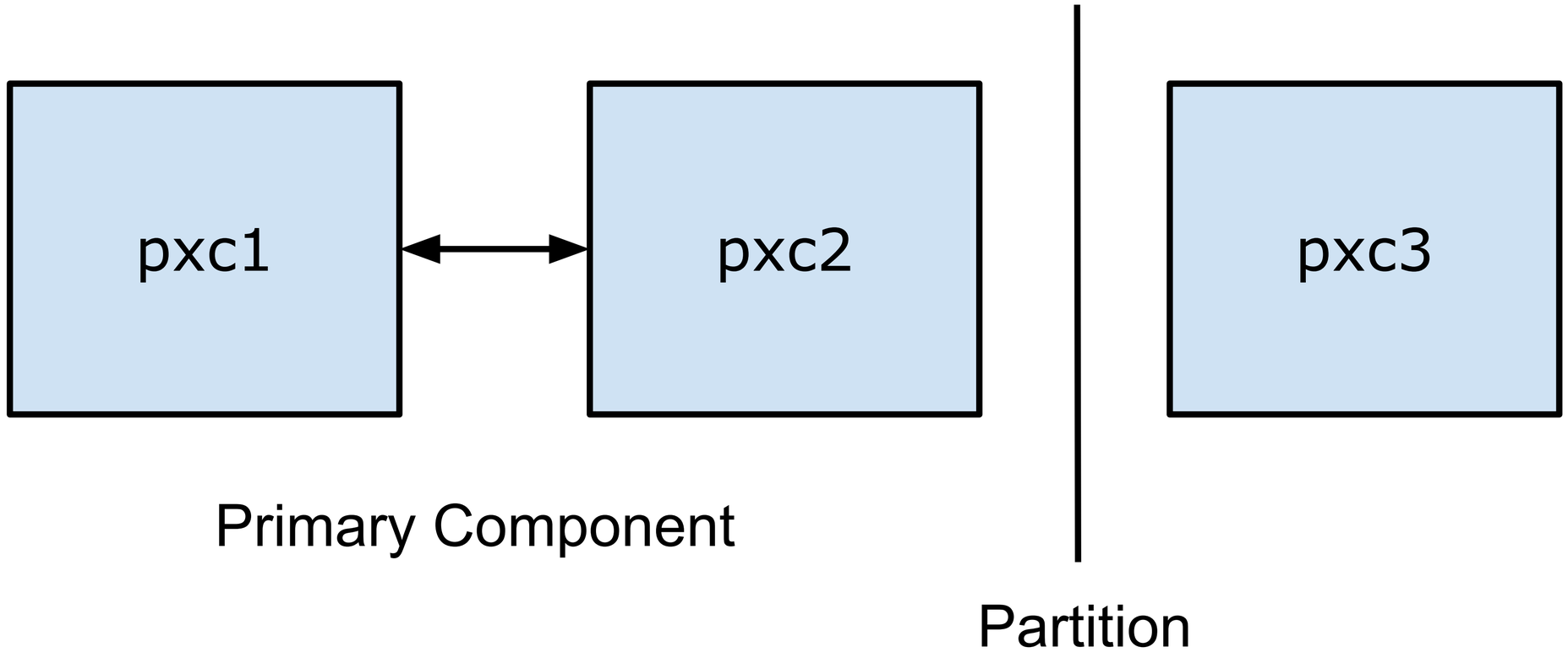
# Synchronous replication

# Parallel replication

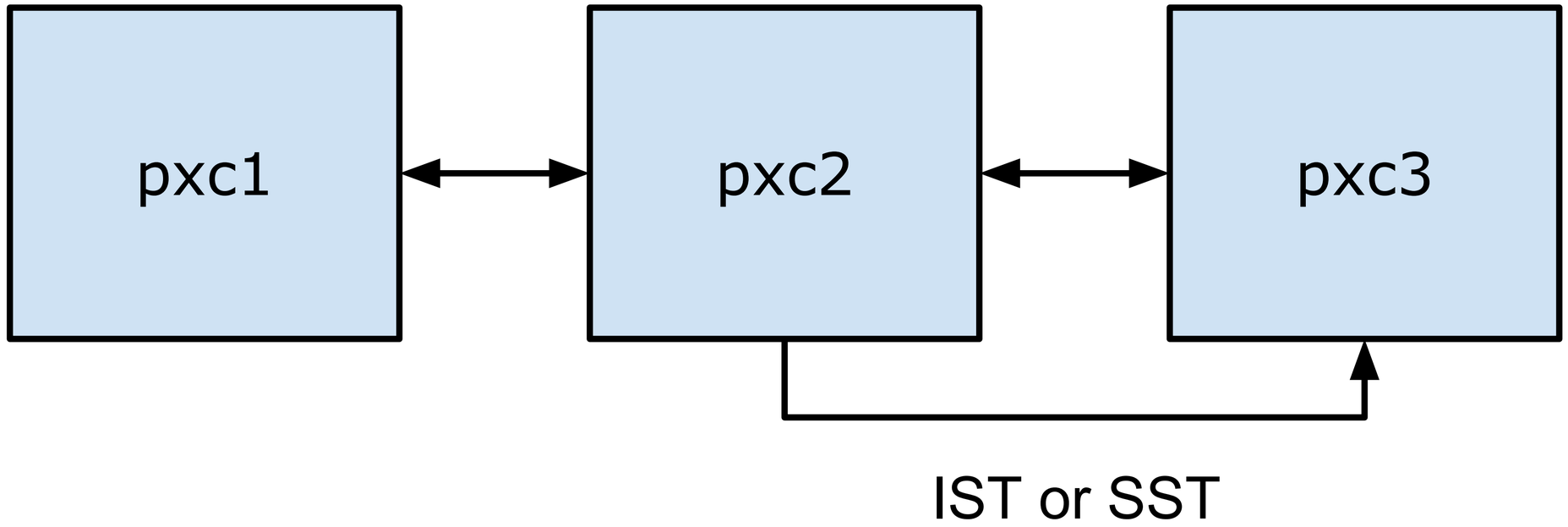
# Automatic provisioning



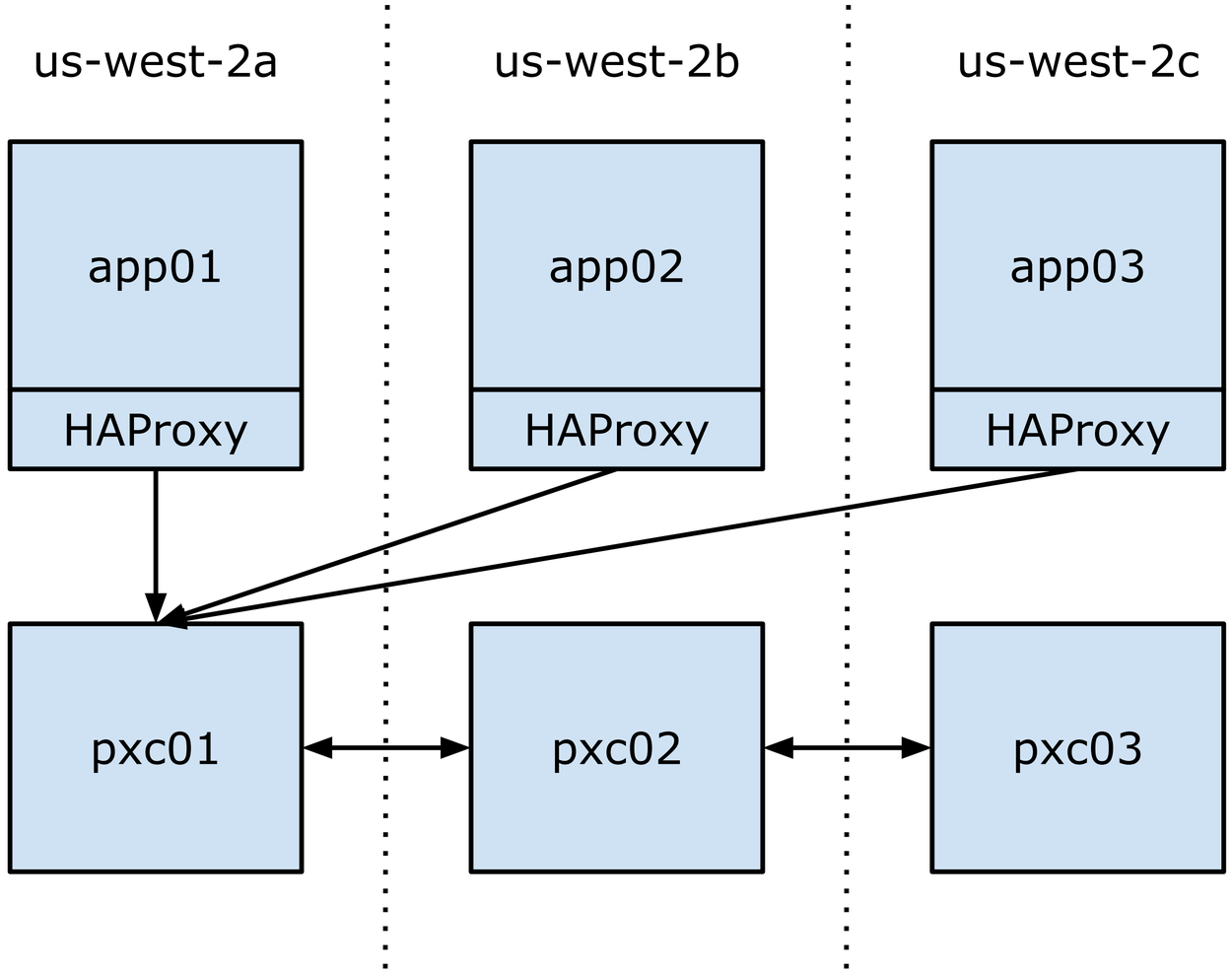
# Failure handling



# Failure handling



# The New Hotness




http://www.mysqlperformanceblog.com/2012/06/20/percona-xtradb-cluster-reference-architecture-with-haproxy/

# SST configuration

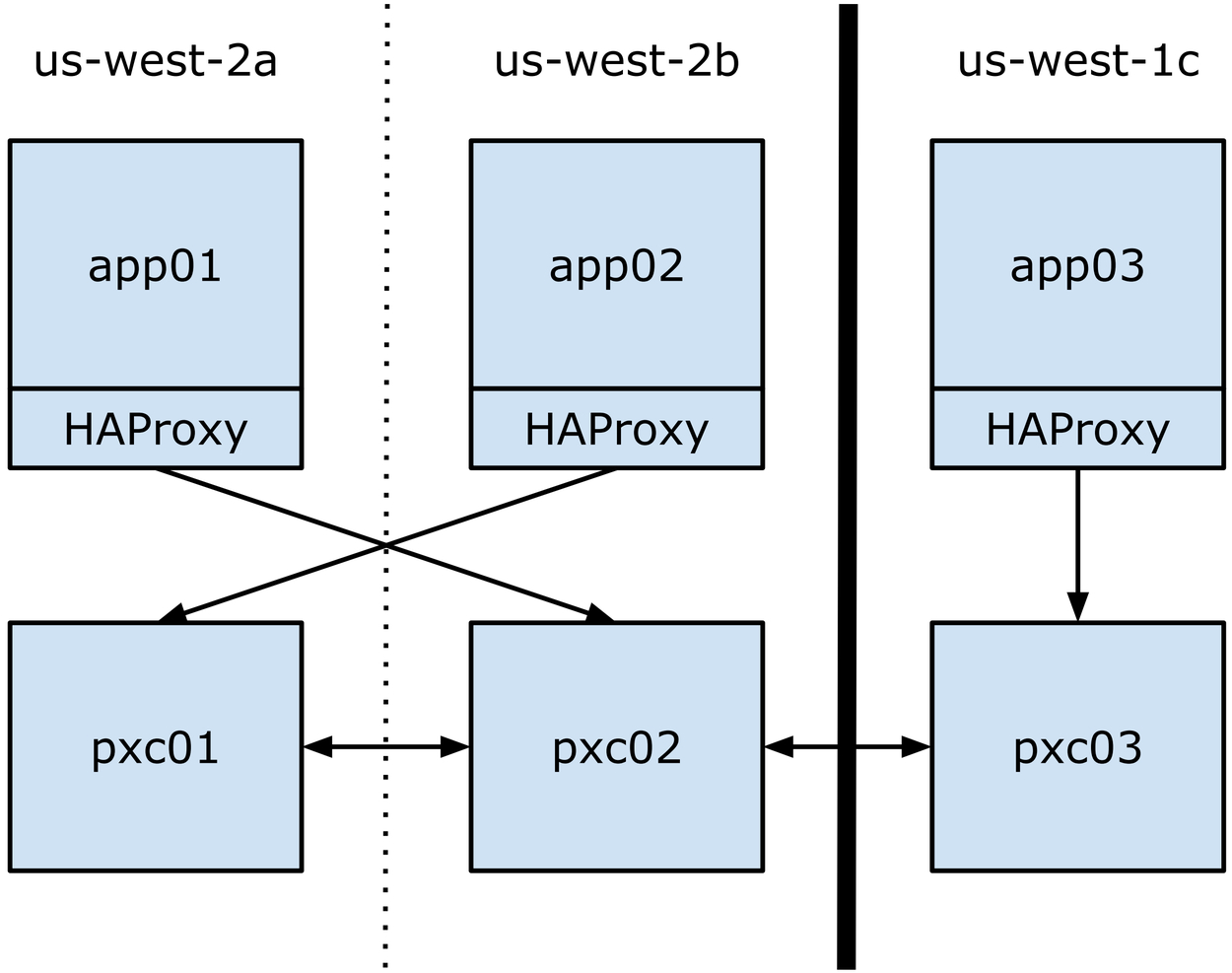
```
wsrep_node_name = pxc01
wsrep_sst_donor = pxc03,pxc02
wsrep_sst_method = xtrabackup-v2
```

# Causal reads enabled

```
wsrep_causal_reads = ON
```

# Buffer pool restoration enabled

```
innodb_blocking_buffer_pool_restore = ON
innodb_buffer_pool_restore_at_startup = 300
```


us-west-2a
us-west-2b
us-west-1c
app01
app02
app03
HAProxy
HAProxy
HAProxy
pxc01
pxc02
pxc03

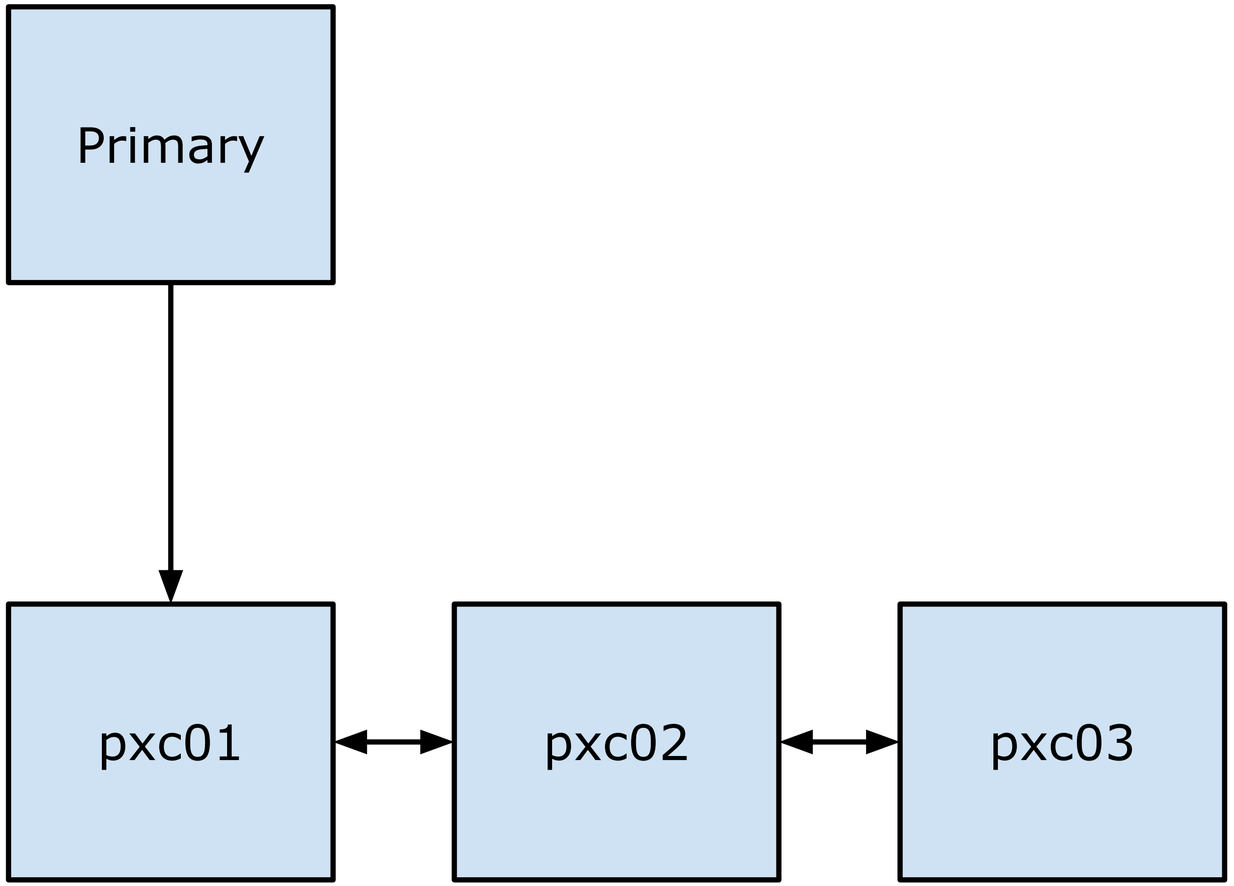
# The Long Road to Production

# Step 1 — Schema compatibility

**InnoDB only**

**Primary keys on every table**

# Step 2 — Gain Experience



# Step 3 — Monitoring


pagerduty

# Step 3 — Monitoring

`wsrep_flow_control_paused`

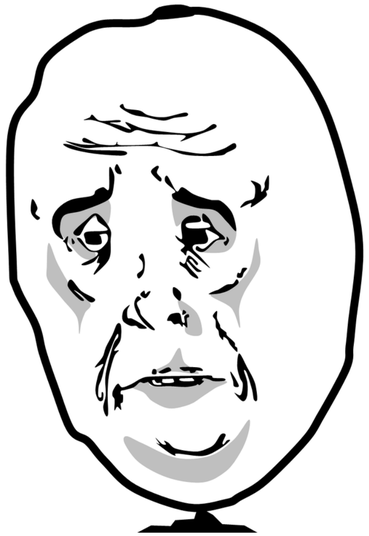
`wsrep_flow_control_sent`

`wsrep_flow_control_received`

# Step 3 — Monitoring

wsrep_flow_control_paused
wsrep_flow_control_sent
wsrep_flow_control_received

Stateful counters in 5.5

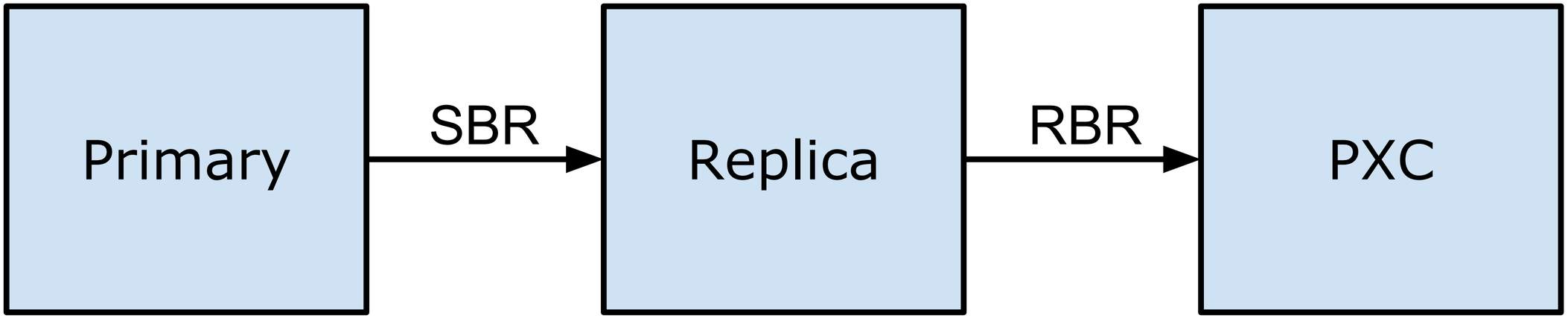


# Step 4 — Query cache not supported

**Measure usage then**

**disable query cache**

# Step 5 — Row base replication



# Step 6 — Locking

**Locking per node**

**Switch to Zookeeper**

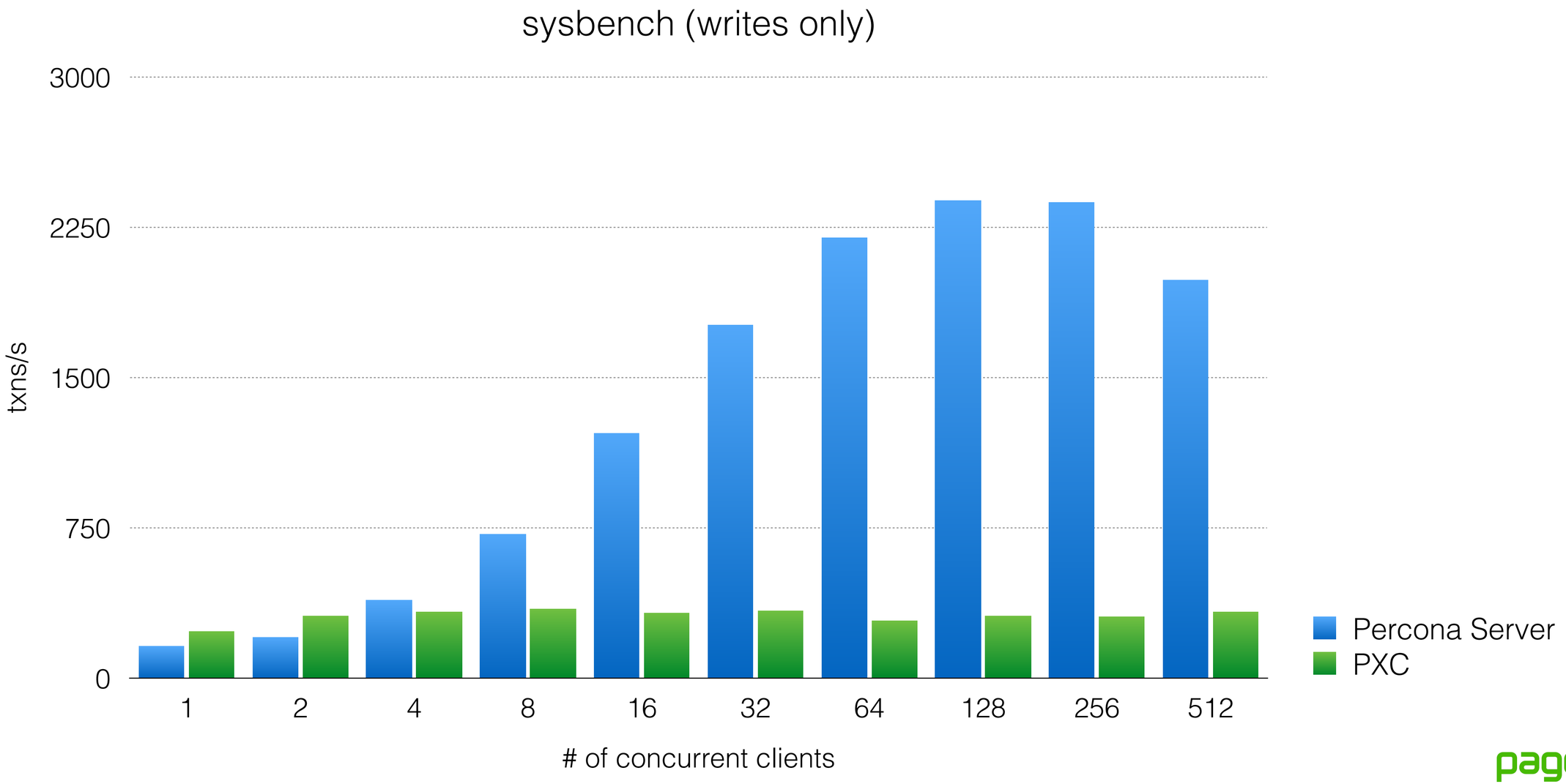
# Step 7 — Eliminate large transactions

**Break up transactions into several smaller ones**

# Step 8 — Benchmarks

**sysbench**

# Step 8 — Benchmarks

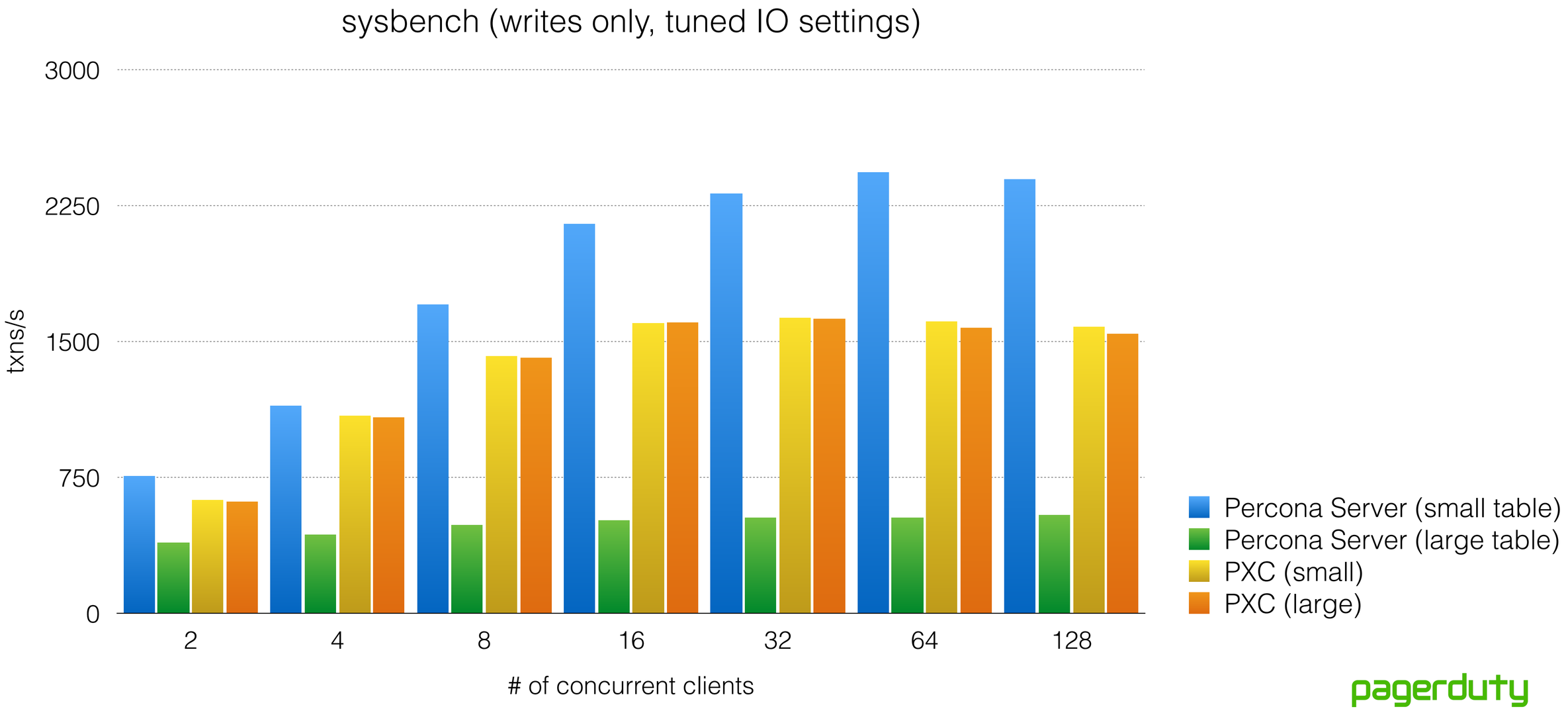



pagerduty

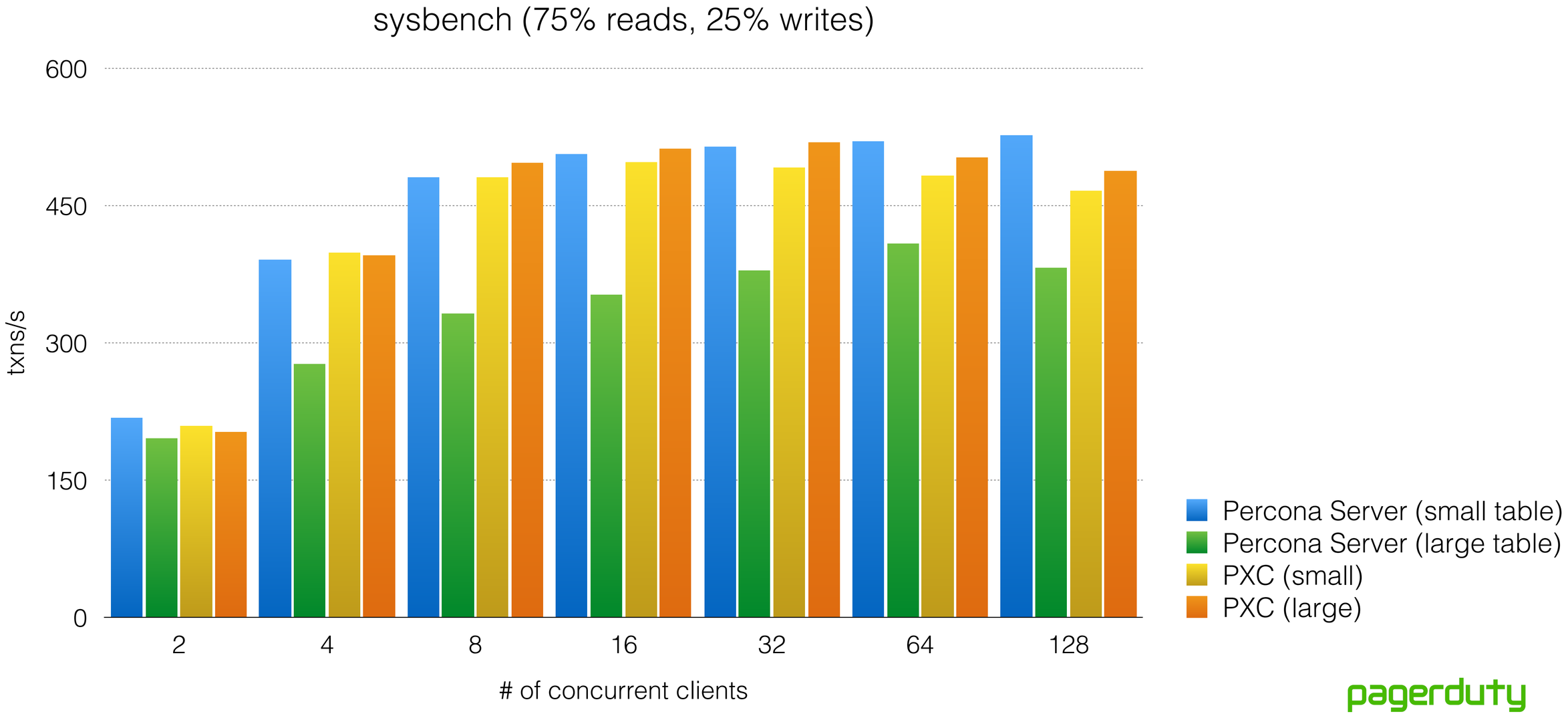
# Step 8 — Benchmarks

```
innodb_flush_log_at_trx_commit = 0
innodb_log_file_size = 1G
```

# Step 8 — Benchmarks



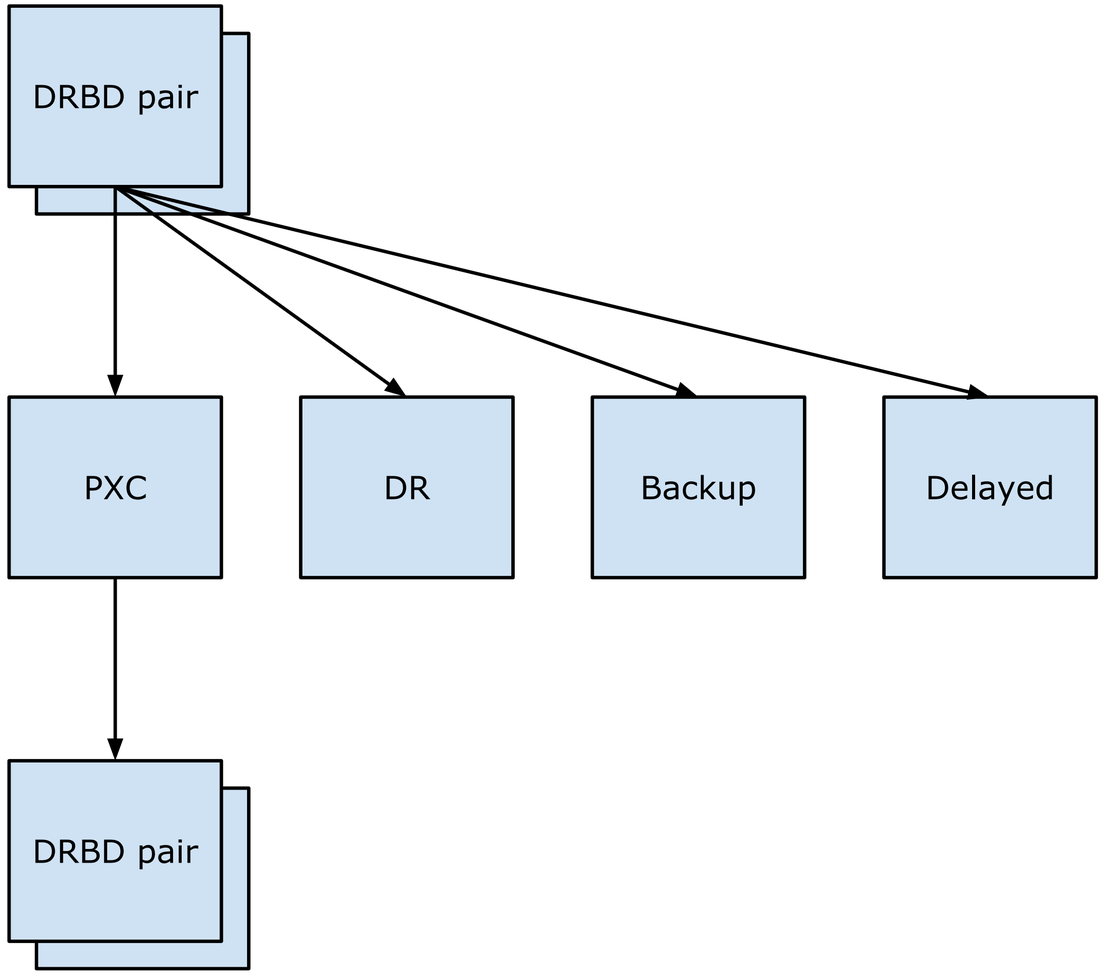
# Step 8 — Benchmarks



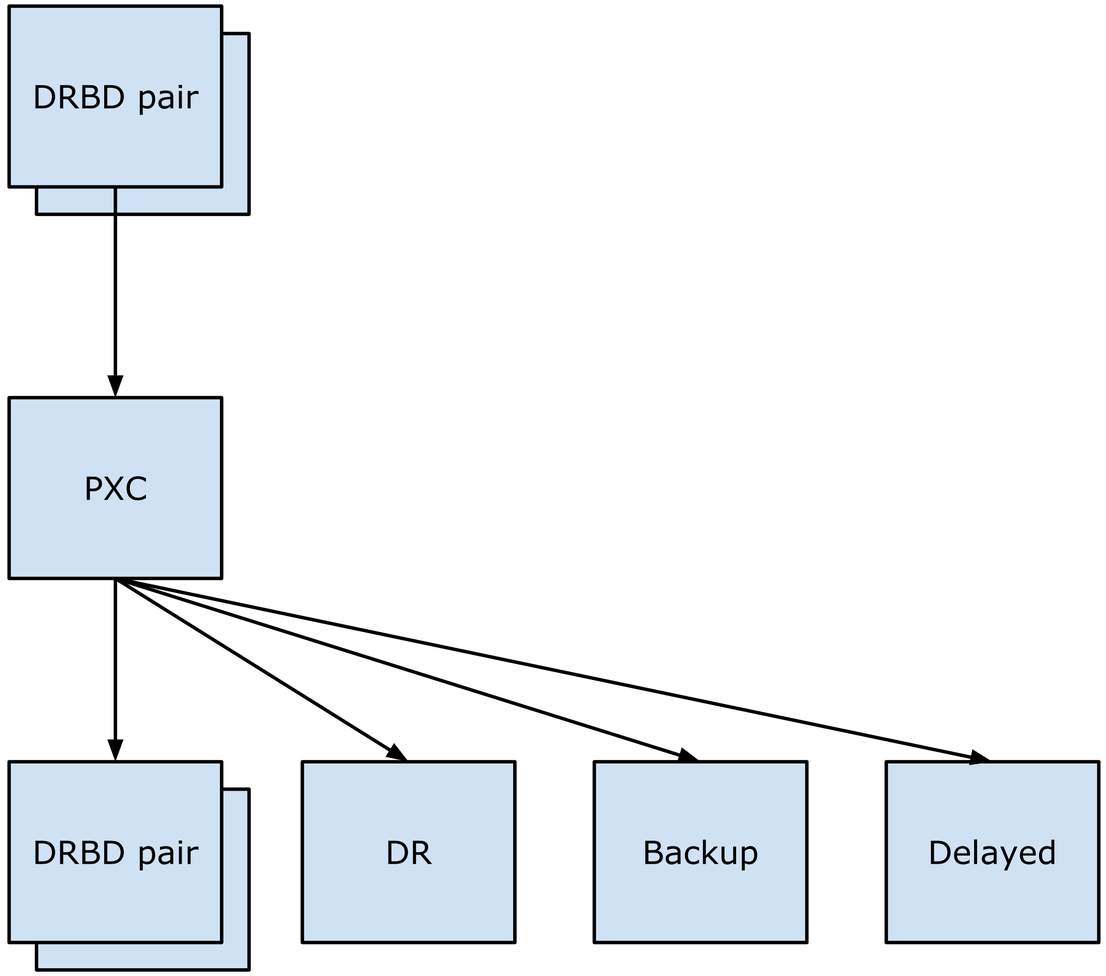
# Production rollout

*Oct 2013*

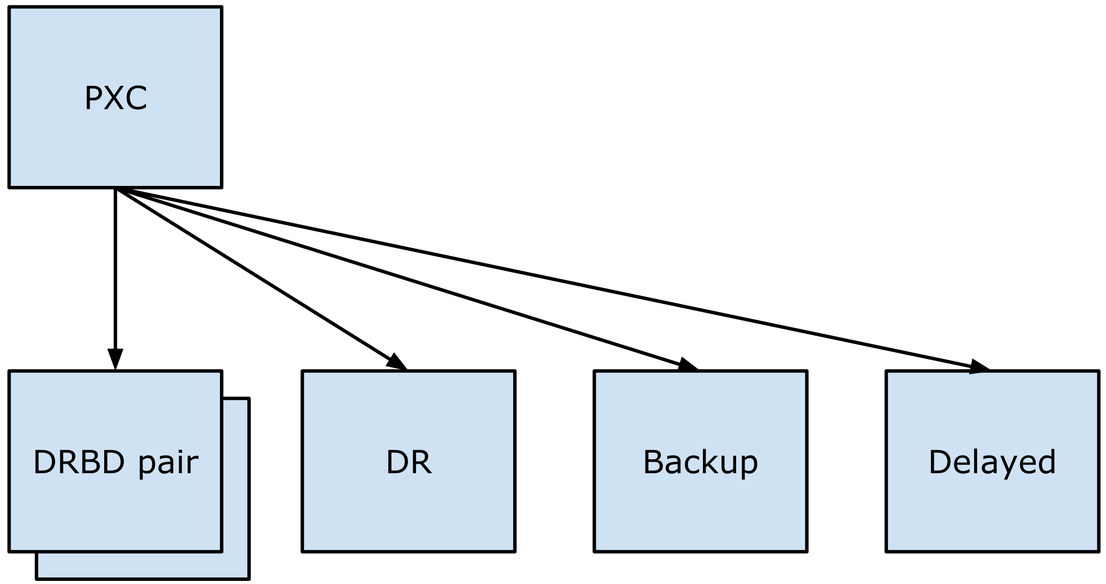
# Rollout



# Rollout



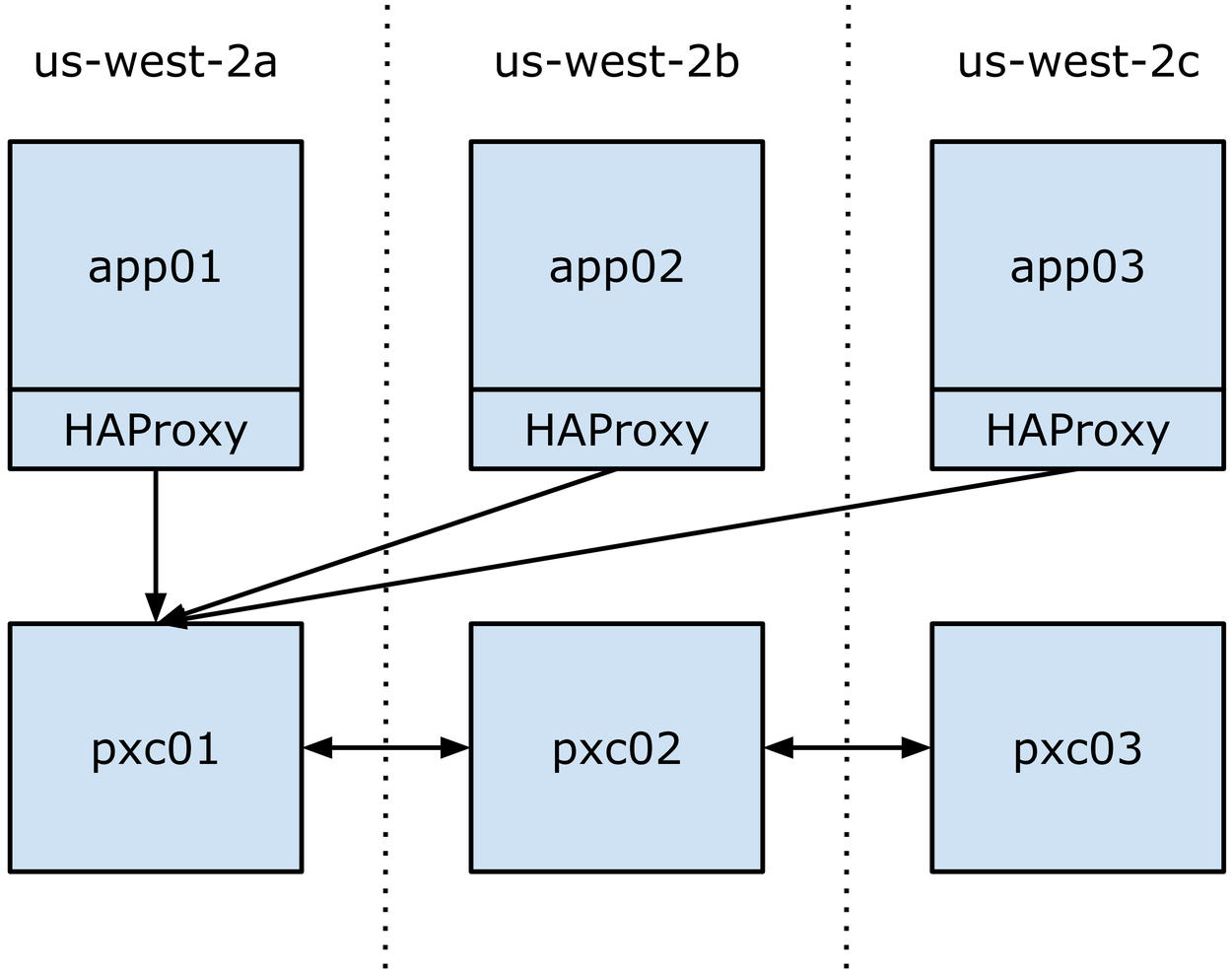
# Rollout



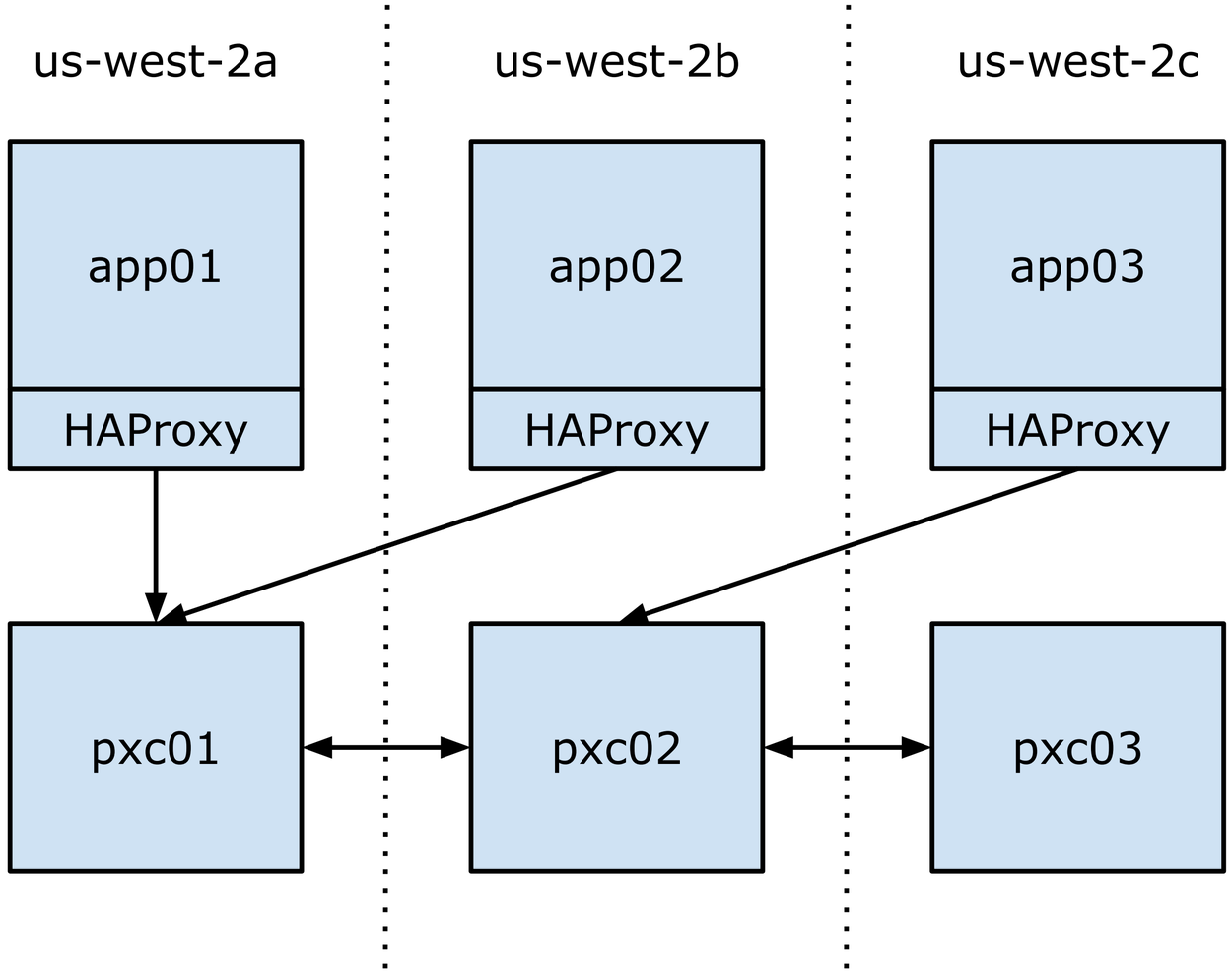
# Life in Production

# Rolling changes

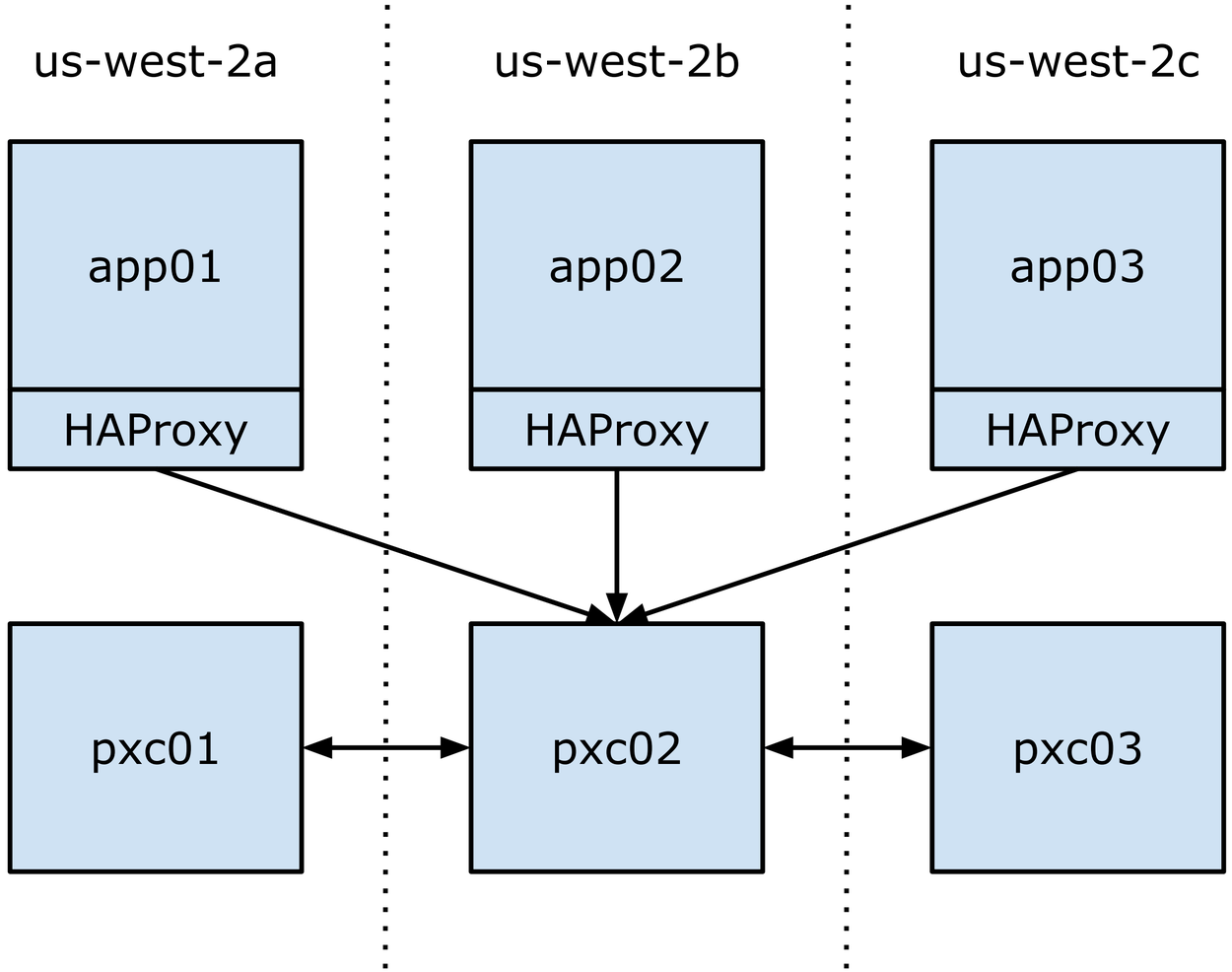
# Benefit: Rolling changes



# Benefit: Rolling changes

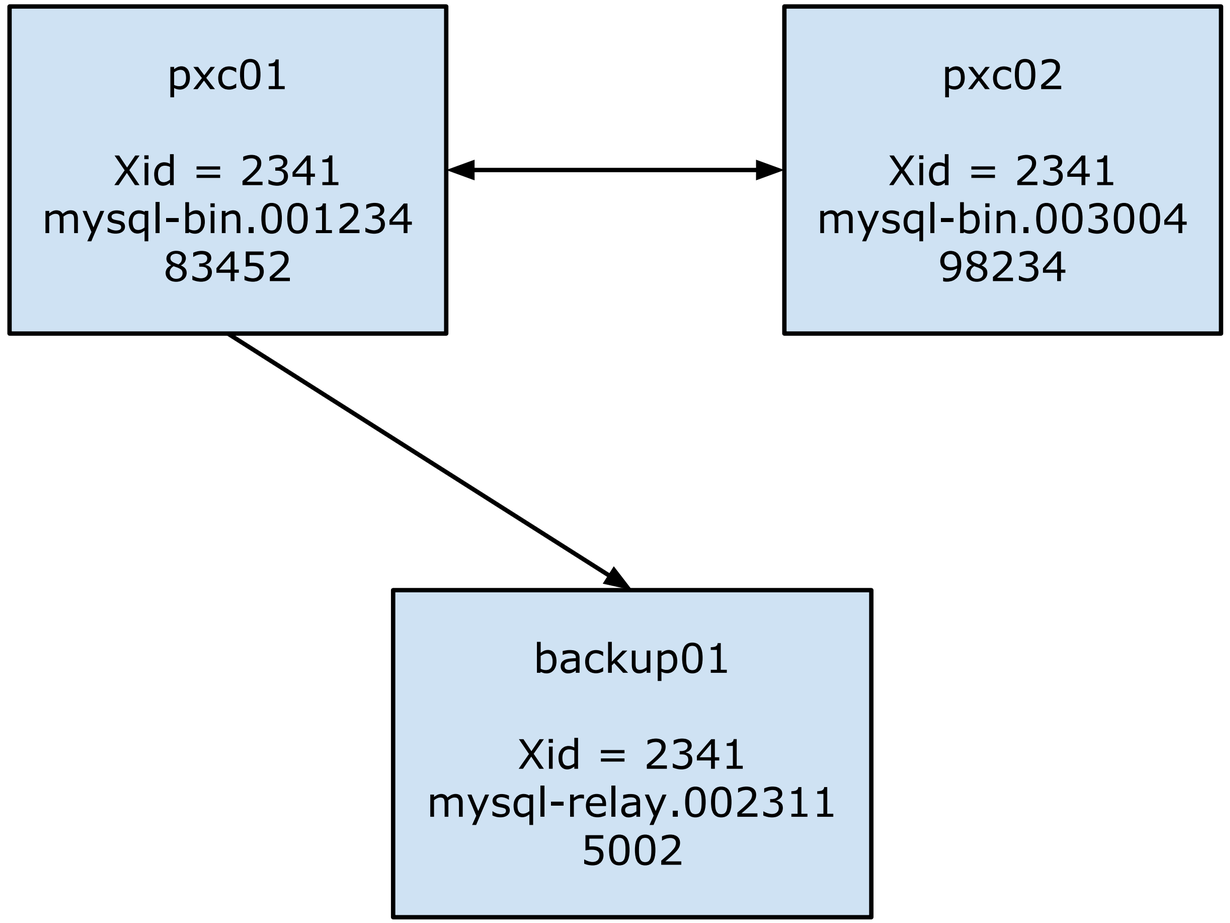


# Benefit: Rolling changes



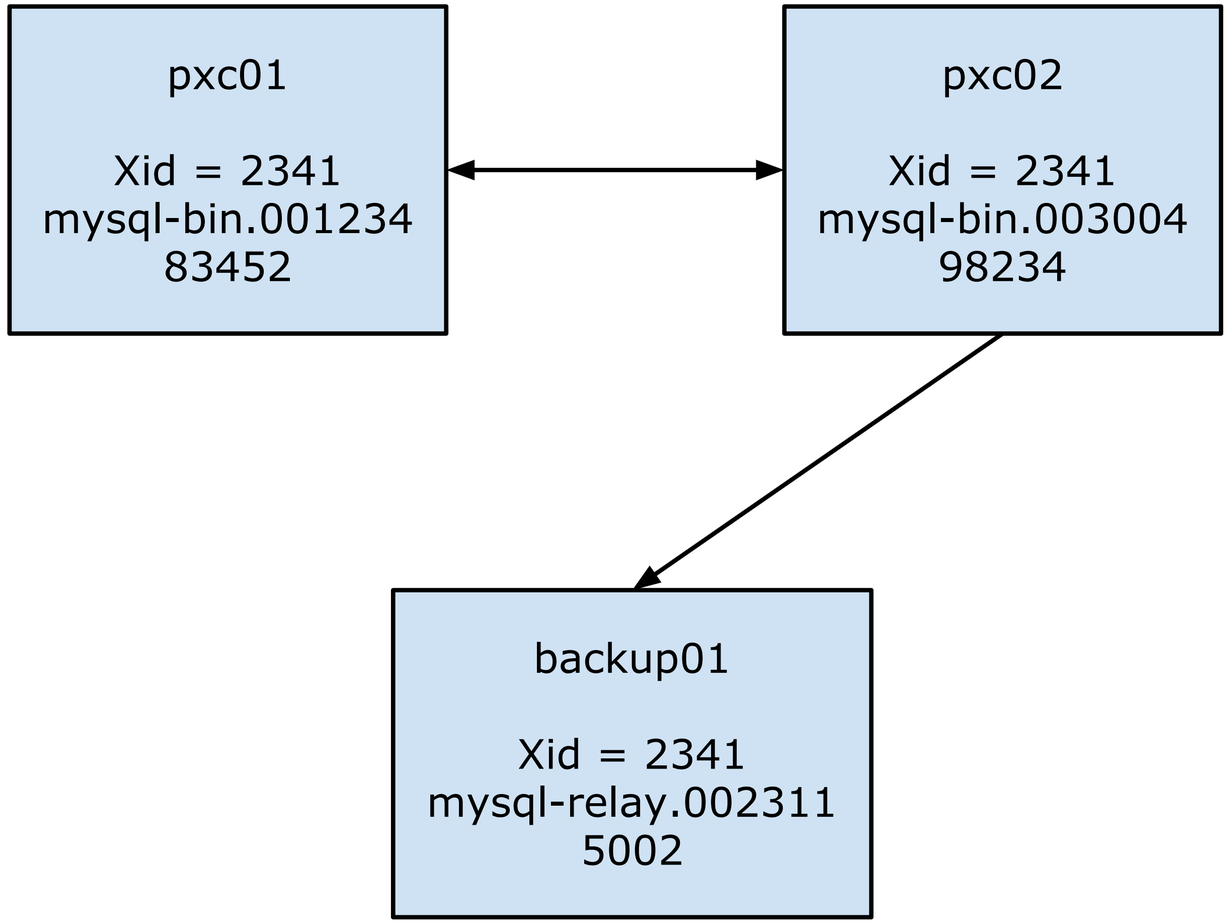
# Moving replicas

# Benefit: Moving replicas




http://www.percona.com/blog/2013/06/21/changing-an-async-slave-of-a-pxc-cluster-to-a-new-master/

# Benefit: Moving replicas

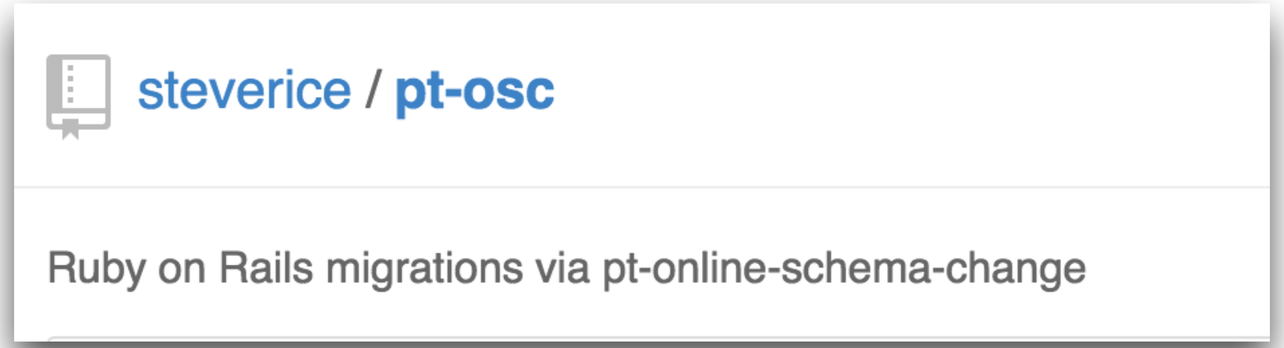



http://www.percona.com/blog/2013/06/21/changing-an-async-slave-of-a-pxc-cluster-to-a-new-master/

# Schema changes

# pt-online-schema-change

# github.com/steverice/pt-osc



# github.com/steverice/pt-osc

## Migrations

To run migrations with `pt-online-schema-change`, you need to explicitly opt them in by changing their parent class to `ActiveRecord::PtOscMigration`. For instance, if your migration was

```
class CreateTeams < ActiveRecord::Migration
```

you should change it to

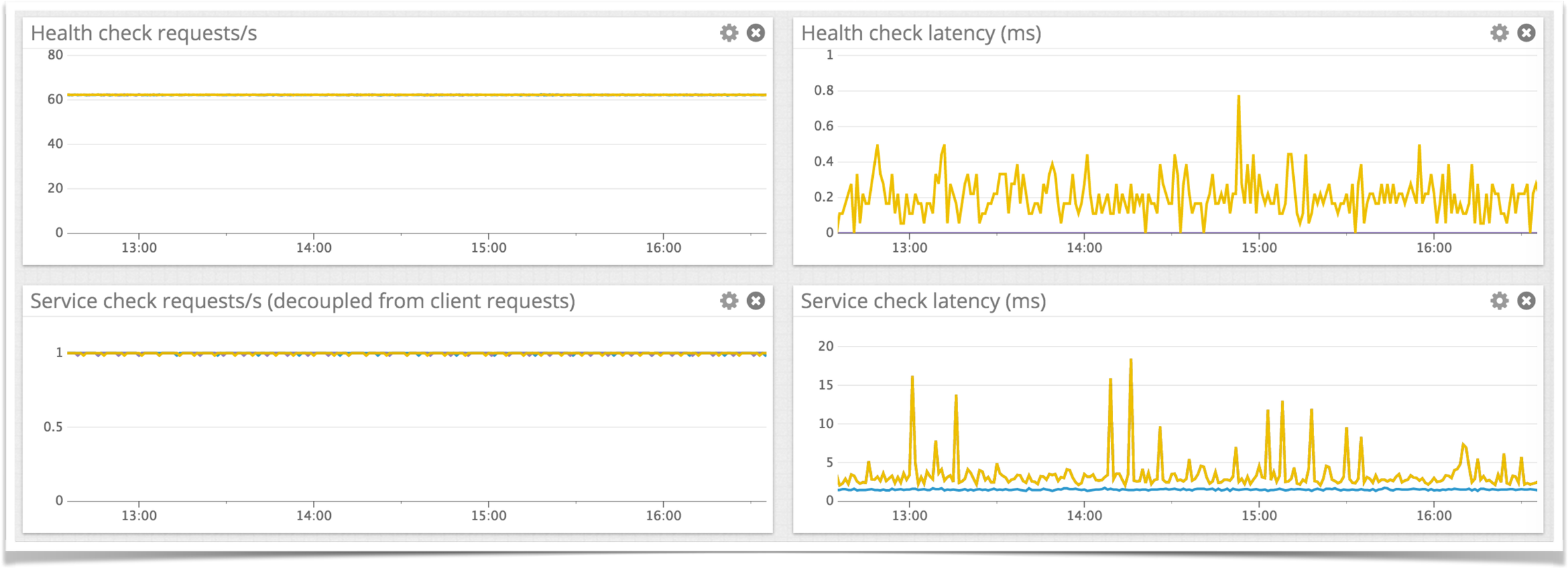
```
class CreateTeams < ActiveRecord::PtOscMigration
```

# github.com/PagerDuty/whazzup

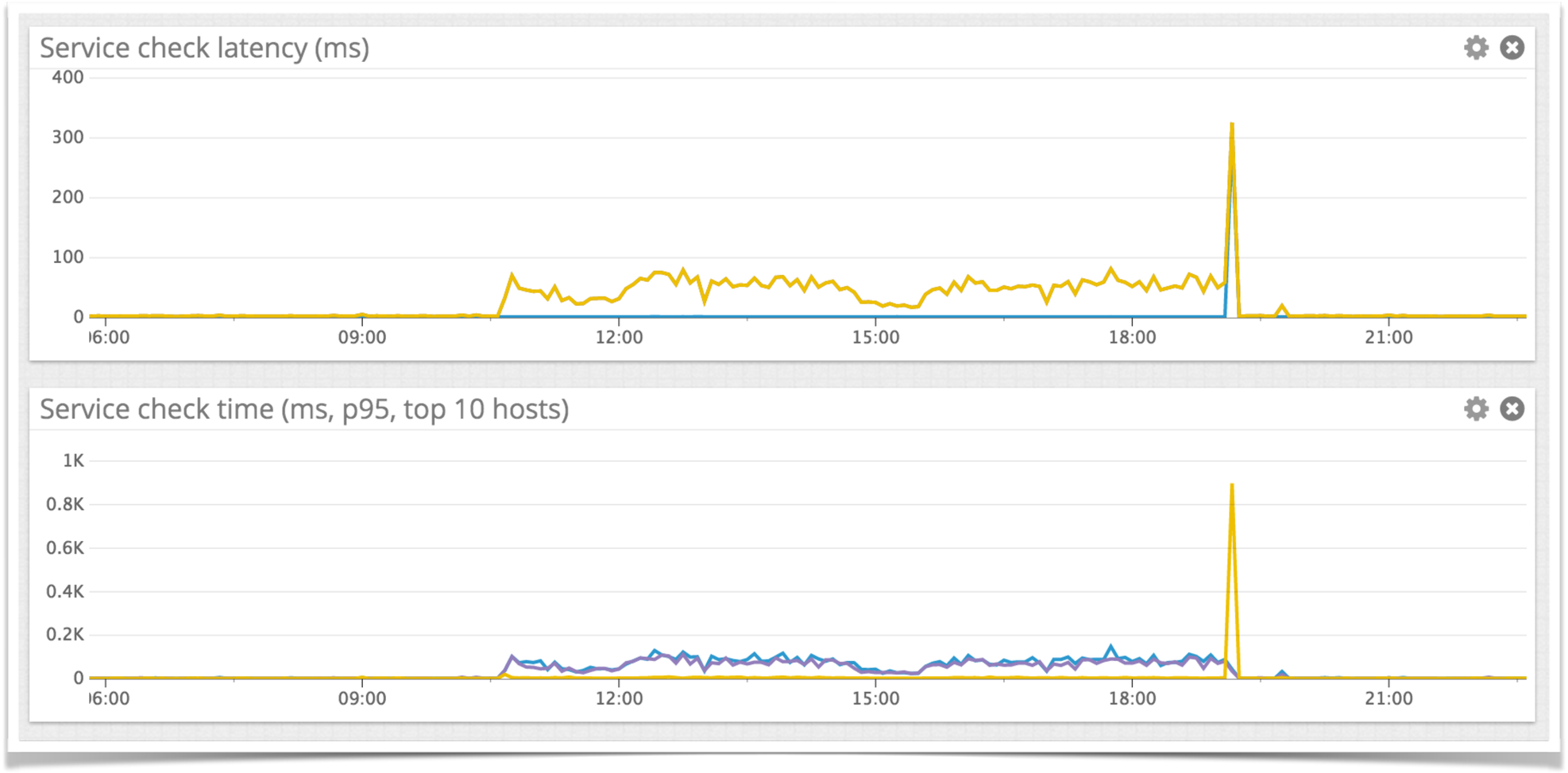
## whazzup

A home for various health check scripts. Health check results are exposed via HTTP, which HAProxy or other clients can hit. The health checks themselves will be rate limited so the number of checks doesn't grow as the number of clients do.

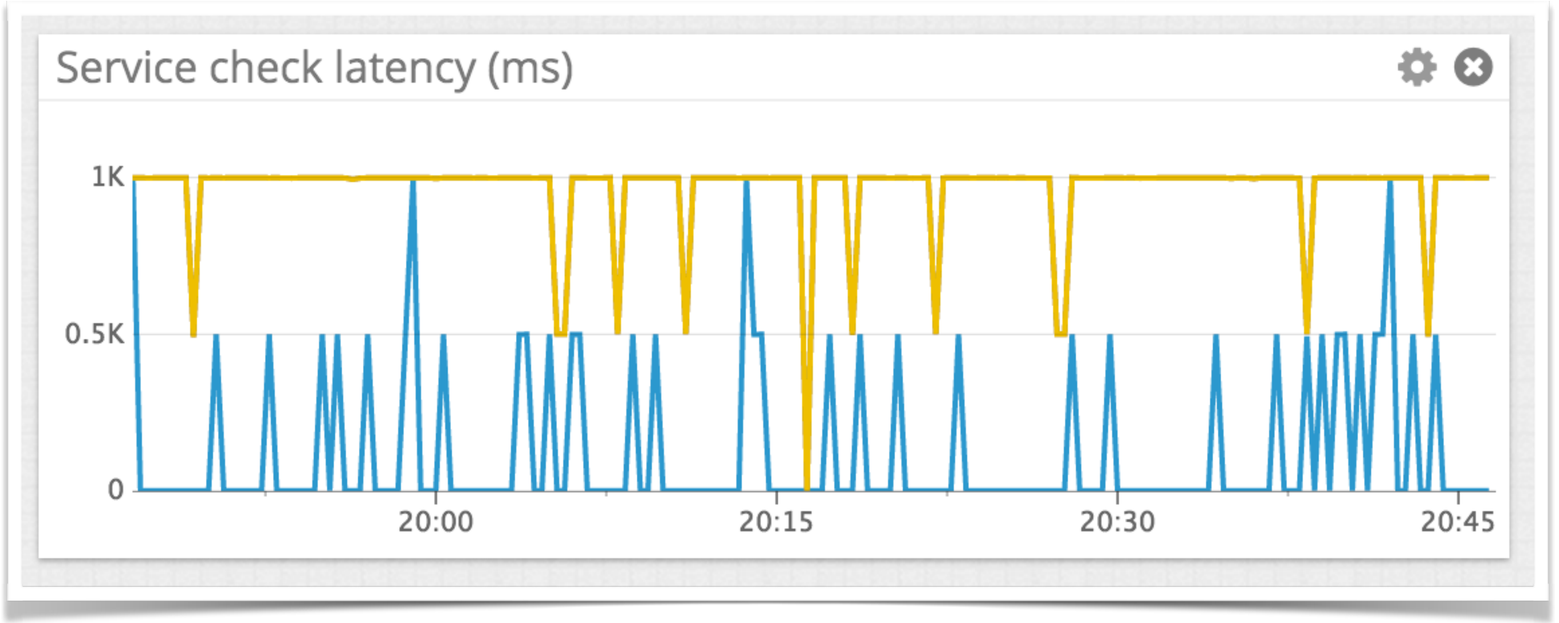
# github.com/PagerDuty/whazzup



# github.com/PagerDuty/whazzup

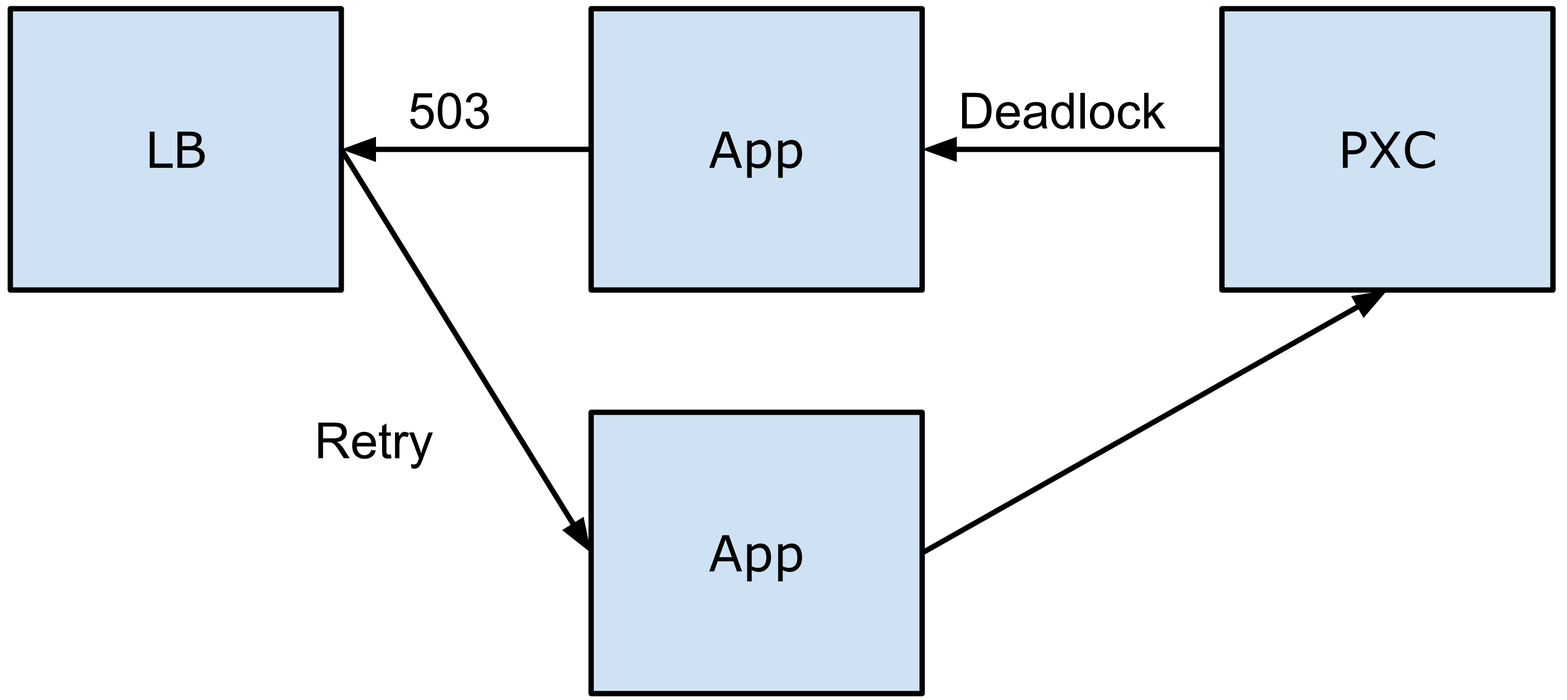


# github.com/PagerDuty/whazzup

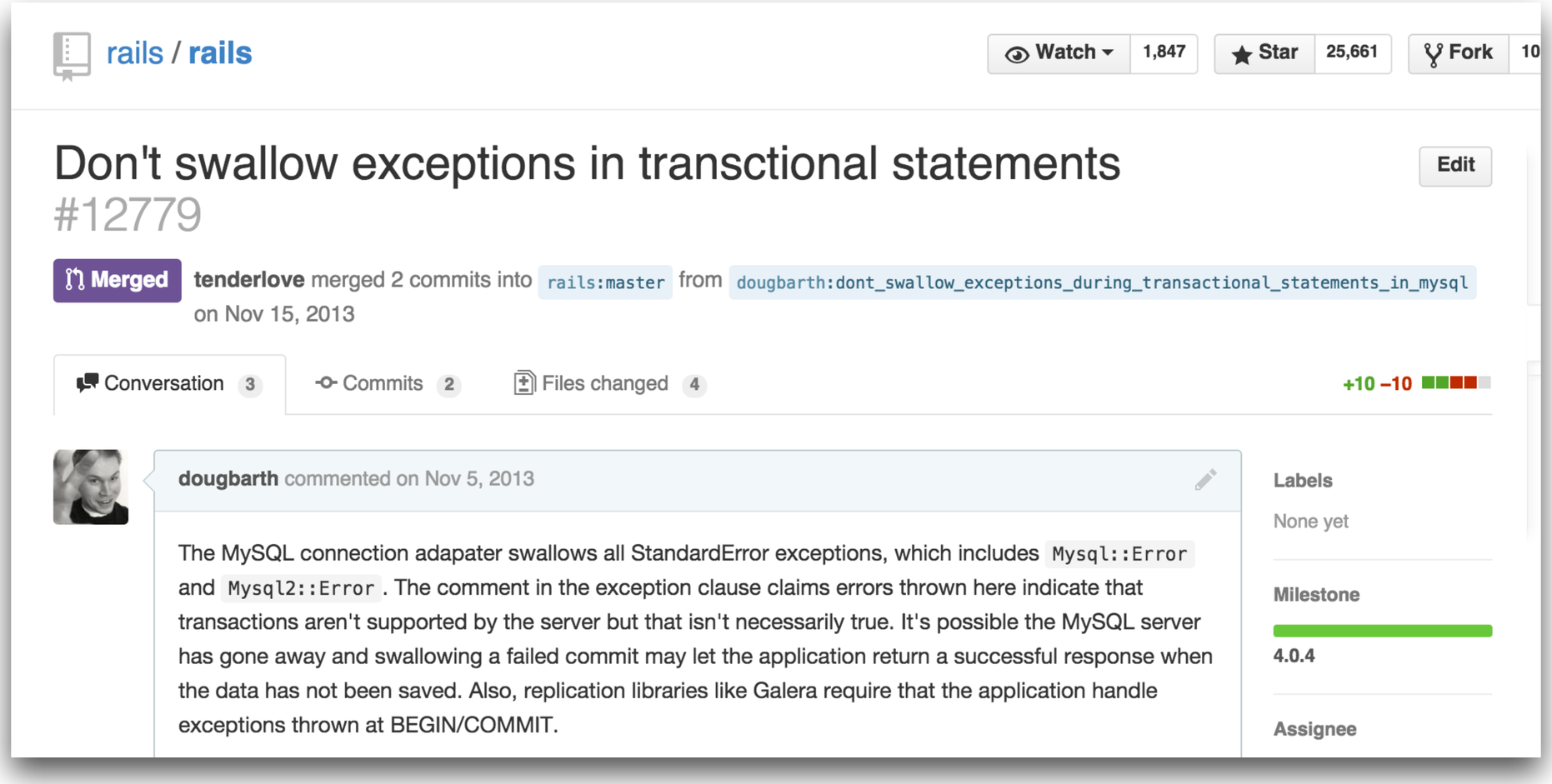



https://bugs.launchpad.net/percona-xtradb-cluster/+bug/1443755

# Deadlock errors



# Deadlock errors




pagerduty

# Failure Friday: How We Ensure PagerDuty is Always Reliable

Kenneth Rose - November 20, 2013

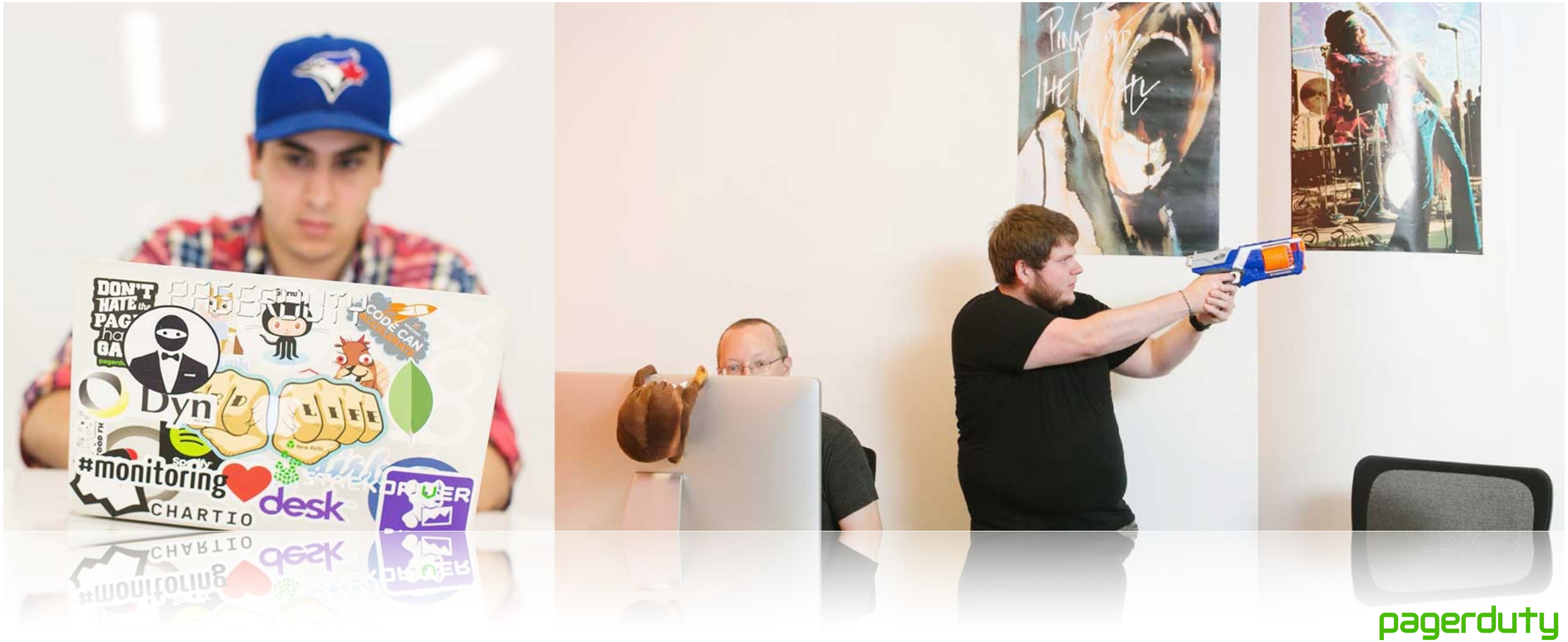
Ask any PagerDutonian what the most important requirement of our service is and you'll get the same answer: Reliability. Our customers rely on us to alert them when their systems are having trouble; on time, every time, day or night.

Our code is deployed across 3 data centers and 2 cloud providers to ensure all phone, SMS, push notifications and email alerts are delivered. Critical paths in our code have backups; and then our code has backups for the backups. This triple redundancy ensures that no one ever sleeps through an alert, misses a text message or doesn't receive an email.

## We Need More Than Failure Tolerant Design

# doug@pagerduty.com

PAGERDUTY.COM/JOBS




pagerduty


# Questions?

pagerduty.com