

# High Availability at Braintree

**Paul Gross**

[paul.gross@braintreepayments.com](mailto:paul.gross@braintreepayments.com)

[twitter.com/pgr0ss](https://twitter.com/pgr0ss)

[github.com/pgr0ss](https://github.com/pgr0ss)

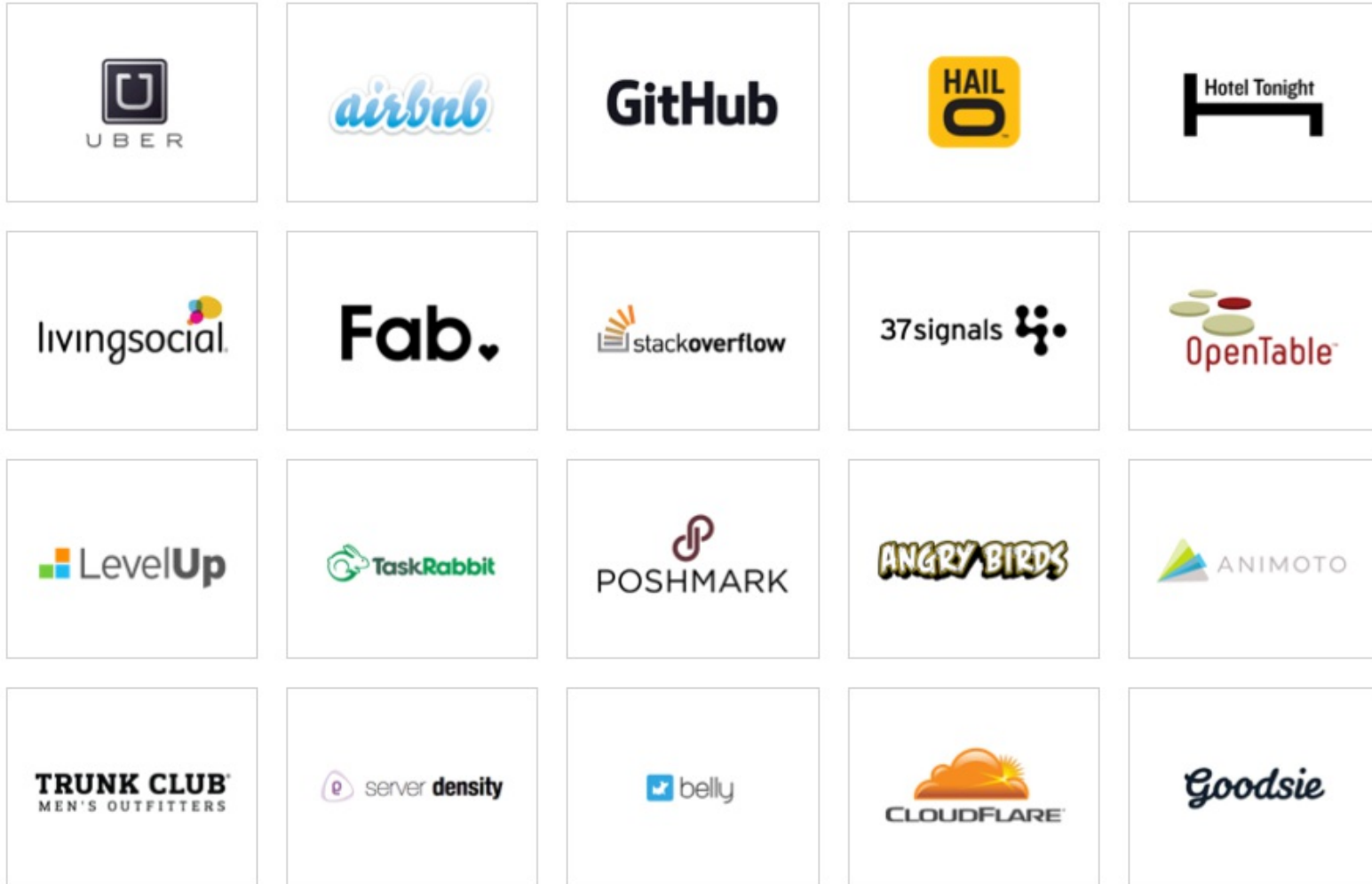
[pgrs.net](http://pgrs.net)

# Braintree

Braintree is a payment gateway

A payment gateway is software that allows merchants to process credit card payments from your website and/or application

# Our Merchants



# Why is uptime important?

\$10 billion in annual processing

\$19,000 per minute for our merchants

# Uptime Percentages

<b>Availability %</b>	<b>Downtime per year</b>	<b>Downtime per month*</b>	<b>Downtime per week</b>
90% ("one nine")	36.5 days	72 hours	16.8 hours
99% ("two nines")	3.65 days	7.20 hours	1.68 hours
99.9% ("three nines")	8.76 hours	43.8 minutes	10.1 minutes
99.99% ("four nines")	52.56 minutes	4.32 minutes	1.01 minutes
99.999% ("five nines")	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% ("six nines")	31.5 seconds	2.59 seconds	0.605 seconds
99.99999% ("seven nines")	3.15 seconds	0.259 seconds	0.0605 seconds

([https://en.wikipedia.org/wiki/High\\_availability](https://en.wikipedia.org/wiki/High_availability))

# 3 causes of downtime

Planned maintenance

Unplanned failure

People mistakes

**Reduce our maintenance windows**

# Code deploys

Ruby on Rails

Put up maintenance page

Deploy code

Run database migrations

Take down maintenance page



# Switched from MySQL to PostgreSQL

DDL migrations are extremely fast

Add indexes without locking tables

Transactional DDL

<http://www.pgrs.net/2011/03/25/migrating-from-mysql-to-postgresql-slides/>

# Refined deploy process

Add new tables and columns

Roll out new code (server by server)

Add indexes

```
namespace :db do
  task :migrate_pre => :environment do
    ActiveRecord::Migrator.migrate "db/migrate_pre"
  end

  task :migrate_post => :environment do
    ActiveRecord::Migrator.migrate "db/migrate_post"
  end
end
```

# Rails caches columns

Can't drop columns in a post migration

Need to tell Rails to forget the column

```
class User < ActiveRecord::Base
  deleted_columns :old_column
end
```

**We run multiple versions of the code at  
once**

Fine for most features

Feature switches to turn on new features at once

# Limitations

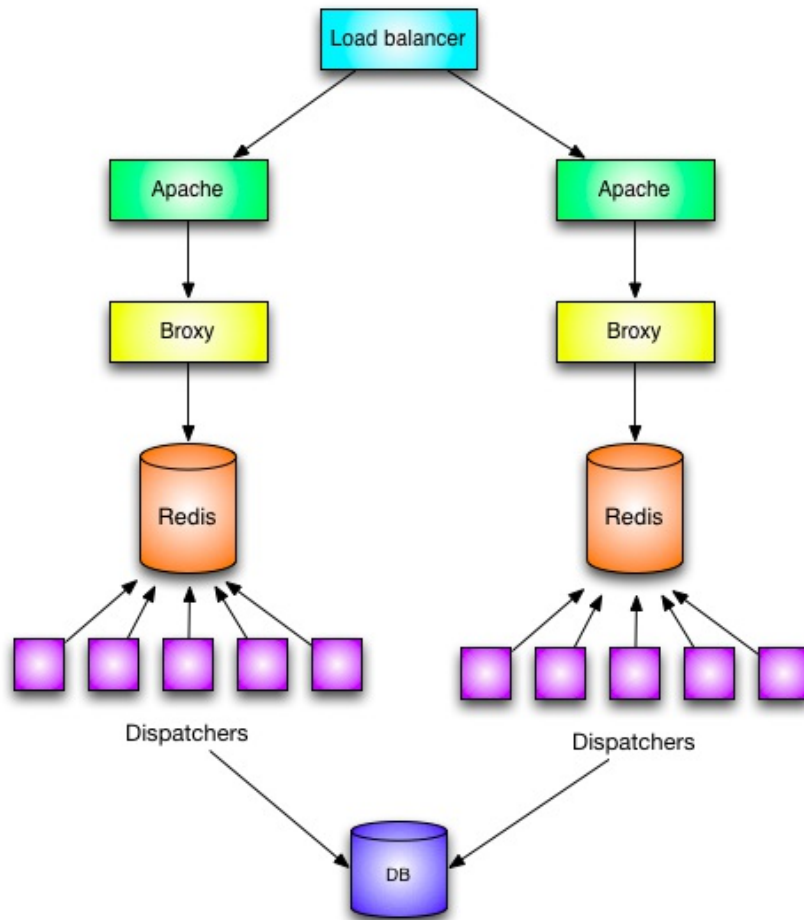
Column renames

Database failover

Infrastructure changes

**Want a way to pause traffic**





# Broxy = Braintree Proxy

Python/Tornado (evented)

Accepts web requests

Feeds redis queue

Reads responses from redis

Responds to the web request

# Dispatchers

Lightweight rack adapter

Takes requests from redis

Processes through rails

Puts response back in redis

```
require "#{ENV['RAILS_ROOT']}/config/environment"  
app = Rack::Builder.new do  
  run ActionController::Dispatcher.new  
end
```

```
require "#{ENV['RAILS_ROOT']}/config/environment"
app = Rack::Builder.new do
  run ActionController::Dispatcher.new
end

loop do
  if request_data = redis.pull_request
    rack_request = {
      "PATH_INFO" => request_data["request"]["uri"]
      "rack.input" => StringIO.new(request_data["body"])
    }

    rack_response = app.call(rack_request)
  end
end
```

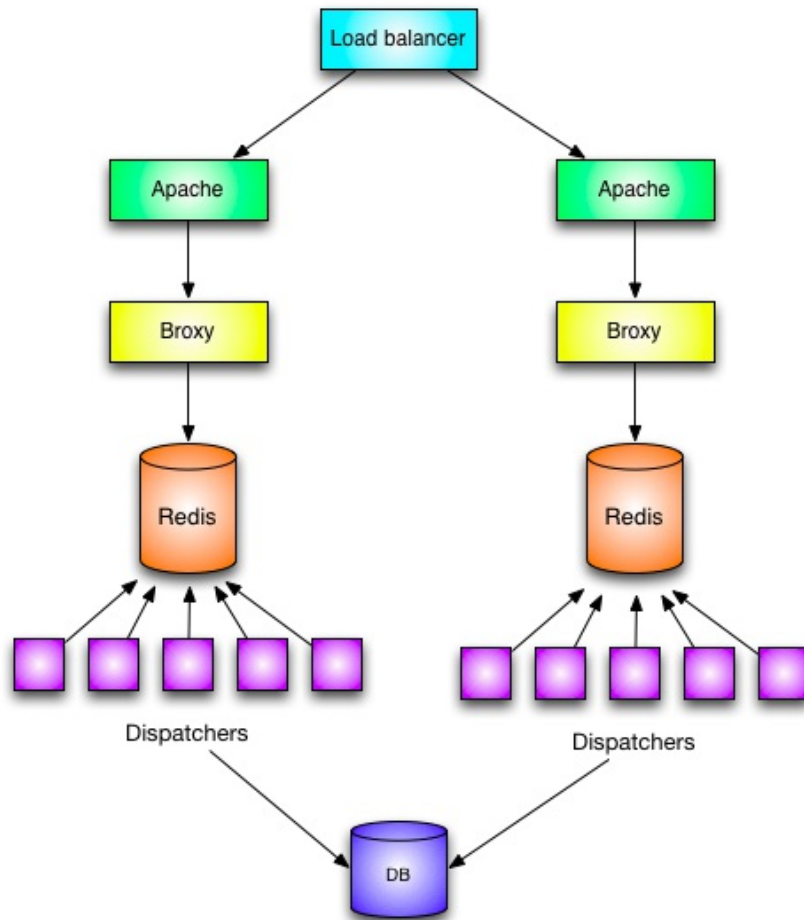
```
require "#{ENV['RAILS_ROOT']}/config/environment"
app = Rack::Builder.new do
  run ActionController::Dispatcher.new
end

loop do
  if request_data = redis.pull_request
    rack_request = {
      "PATH_INFO" => request_data["request"]["uri"]
      "rack.input" => StringIO.new(request_data["body"])
    }

    rack_response = app.call(rack_request)

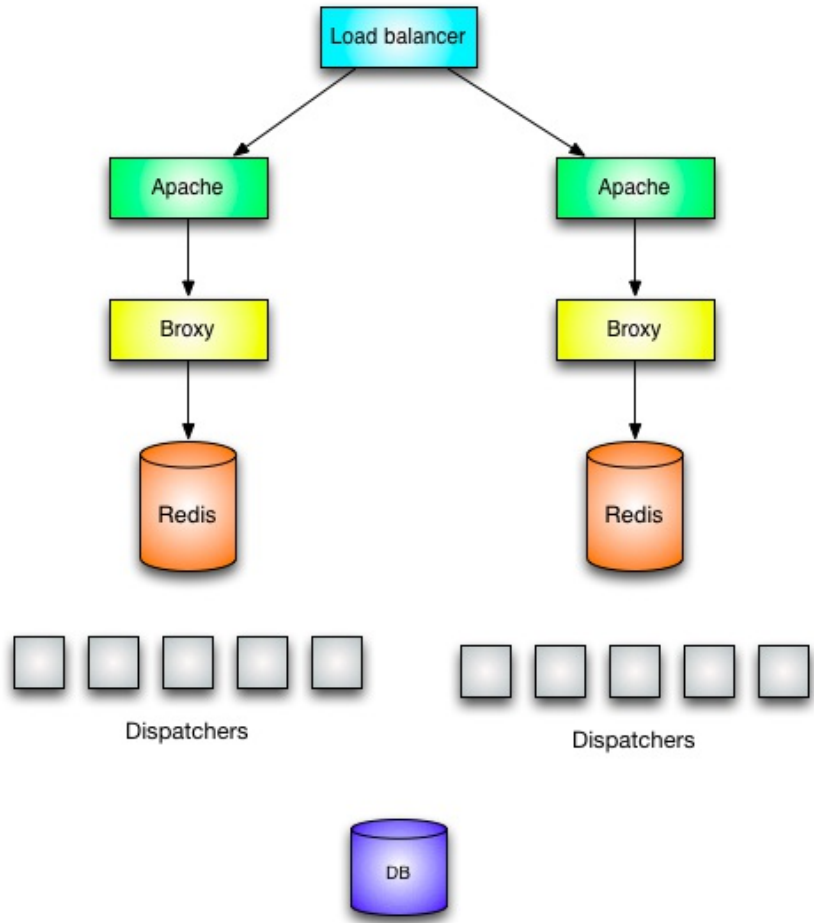
    body = ""; rack_response[2].each { |part| body << part }

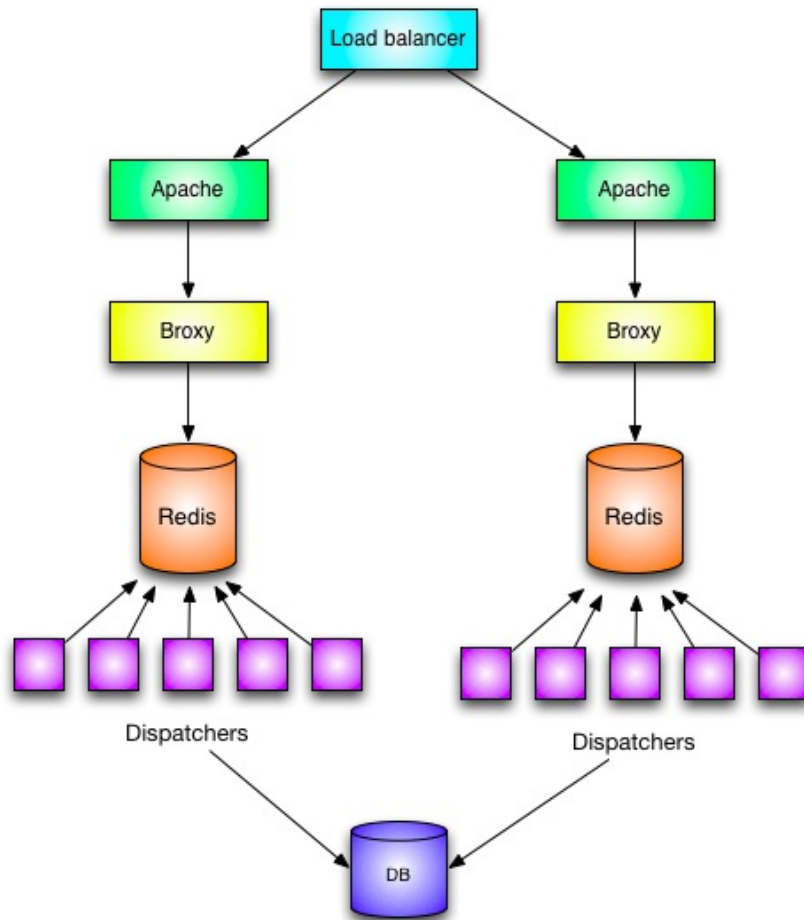
    @redis.push_response(
      "status" => rack_response[0].to_i,
      "body" => body
    )
  end
end
```



**Stop dispatchers to suspend traffic**







# Summary - reducing maintenance windows

Pre and post migrations

Rolling deploys

PostgreSQL for fast DDL

Broxy to pause traffic

# Unplanned failures

Servers will fail

Networks will go down

The unexpected will happen

We do our best to be resilient

**Server failure**

# Load balancers

Build our own

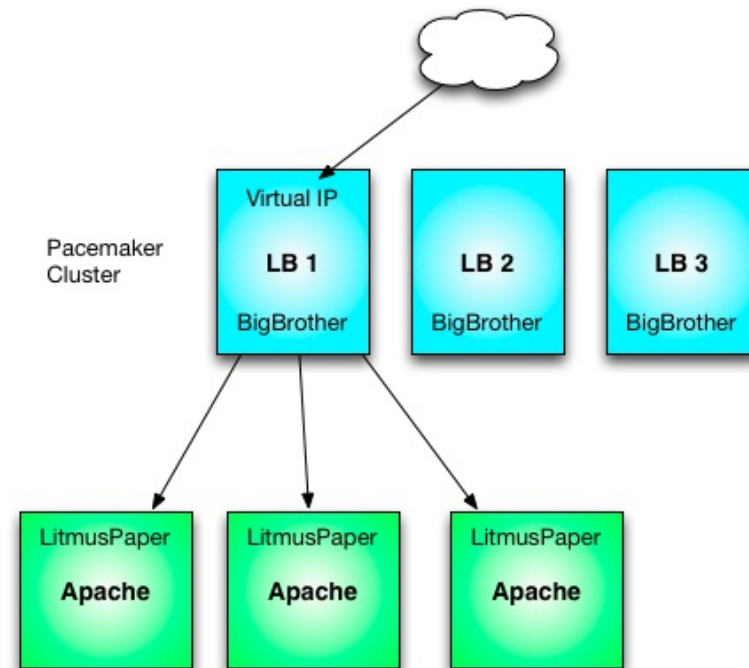
LVS/IPVS

Pacemaker

BigBrother

LitmusPaper

# Load balancing



# BigBrother

Ruby app

Runs on load balancers

Checks status of servers

Update IPVS rules

[https://github.com/braintree/big\\_brother](https://github.com/braintree/big_brother)



# LitmusPaper

Ruby app

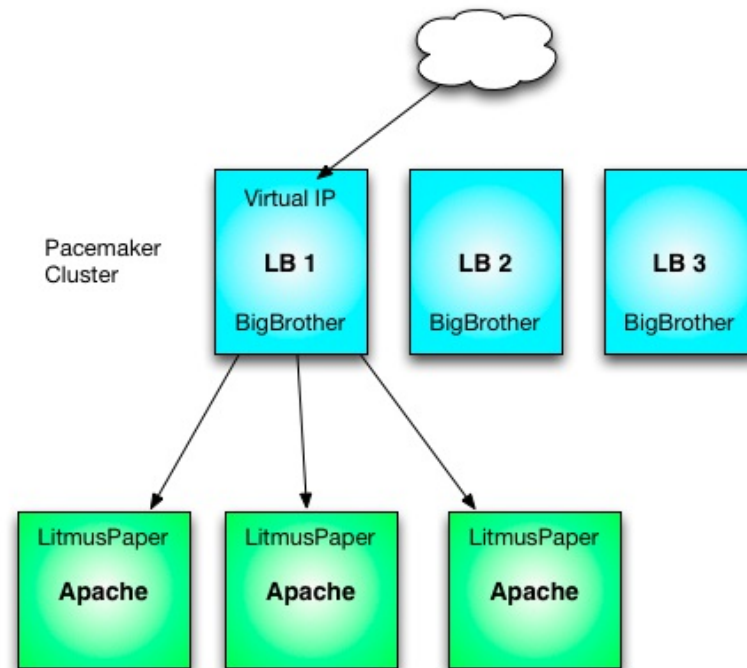
Runs on backend servers

Queried by BigBrother via HTTP

Returns a health level

[https://github.com/braintree/litmus\\_paper](https://github.com/braintree/litmus_paper)

# Load balancing



# Stateful services

Load balancers and PostgreSQL clusters

Pacemaker manages failover

Virtual IP follows the new primary

# Network failures

Internet Health Report

internetpulse.com

keynote The Mobile & Internet Performance Authority™

Internet Health Report Generated from 9/16/2013 7:34:59 PM to 9/16/2013 8:34:59 PM (PDT)

Monitor your performance | Help

Focus: From: AT&T To: AT&T Metric: Latency (ms) Period: Last 1 Hour

View: Destination by Origin Metrics by Origin

Destination - Latency (ms) - Last 1 Hour

	AT&T	CenturyLink	Cogent	Level3	NTT	Savvis	SBC	Sprint	Verizon	XO
AT&T	4	26	112	65	27	70	25	74	65	41
CenturyLink	25	15	87	53	94	45	40	44	53	48
Cogent	109	93	15	51	54	17	69	19	60	28
Level3	72	56	54	1	71	15	29	24	33	27
NTT	27	94	55	69	1	77	48	83	122	43
Savvis	69	39	17	21	76	7	32	19	33	25
SBC	28	42	63	28	52	25	12	37	39	25
Sprint	74	45	18	26	83	18	37	2	40	33
Verizon	67	52	73	32	118	35	39	41	11	40
XO	38	46	28	28	43	23	24	35	40	17

Healthy < 90ms Latency. Warning < 180ms Latency. Critical > 180ms Latency.

Copyright © 2013 Keynote Systems, Inc. All rights reserved. | Terms of Service | Privacy Policy | Site Map | Support

# Networking - inbound

BGP routes traffic through multiple ISPs and data centers

We use Pingdom and a handful of globally distributed servers to test connectivity

# Networking - outbound

We connect to many processing networks

ISP outages are usually partial

Sometimes, we can't reach every endpoint on all of our ISPs

Needed a way to choose an ISP per processing network

# Processor proxies

Instead of connecting directly, connect through proxies

One proxy per TCP endpoint and uplink ISP

Load balance over these proxies

Allows us to route around ISP connection issues



# Mallory

Python/Tornado (evented)

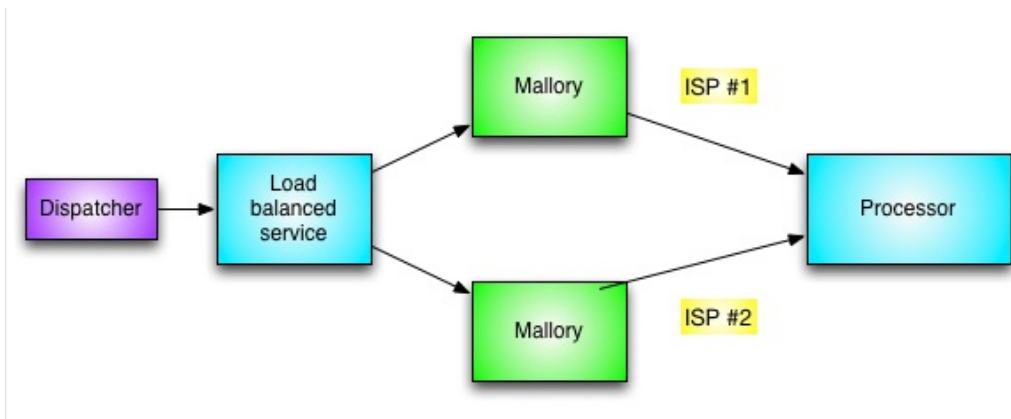
Proxies requests

SSL verification

Acts like LitmusPaper

<https://github.com/braintree/mallory>

# Processor proxies



# Connection failures

Let the service heal (unbalance or pacemaker)

Retry request

# Summary - unplanned failures

Load balancing

Redundancy across ISPs

Let the system heal and retry

# People make mistakes

They can cause outages

They can exacerbate existing issues

# EC2 Outage Apr 21, 2011

"The traffic shift was executed incorrectly and rather than routing the traffic to the other router on the primary network, the traffic was routed onto the lower capacity redundant EBS network."

<https://aws.amazon.com/message/65648/>

# GitHub Outage Nov 14, 2010

"A few hours ago I was upgrading our continuous integration setup when a configuration error caused it to run against our production environment rather than our testing environment."

<https://github.com/blog/744-today-s-outage>

# PagerDuty Outage Jan 26, 2013

"The reason the flip did not work was because the data snapshot on the new machine was not uploaded correctly, due to the engineers being extremely tired and burned out after working through the night on the upgrades."

<http://blog.pagerduty.com/2013/01/outage-post-mortem-jan-24-2013/>



**Reduce the people factor**

# Automate everything

Reduces errors

Gives confidence that task will work

Speeds up processes

Less fiddling around in production

Capistrano and Puppet

# Safeguards

Dangerous tasks refuse to run in production

Prompts include environment name

Read only database and logs

# Monitoring

Nagios

PagerDuty

Multi-app log tail (with sampling)

Graphite and munin

# Summary - people

Automation

Safeguards

Monitoring

# Questions?

**Paul Gross**

[paul.gross@braintreepayments.com](mailto:paul.gross@braintreepayments.com)

[twitter.com/pgr0ss](https://twitter.com/pgr0ss)

[github.com/pgr0ss](https://github.com/pgr0ss)

[pgrs.net](http://pgrs.net)

