Engineering a Modern Web Application

Dave King
James Madison University
Sept 11th, 2013

About Me


Graduate School:

- Publications in USENIX, ICSE, FSE, ESOP
- PhD (2009): “Retrofitting Programs for Complete Security Mediation”

Rackspace (2009-today):

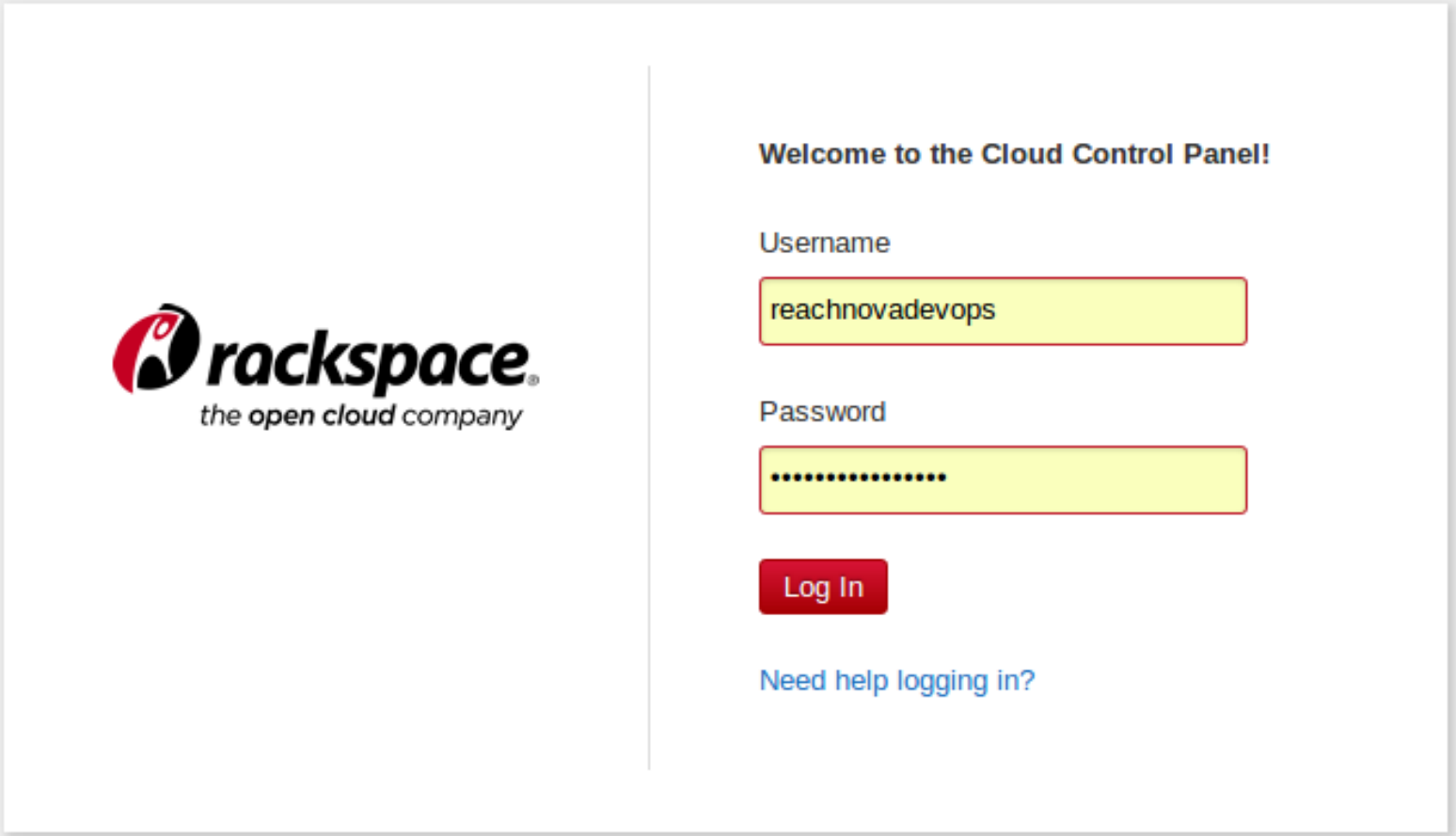
- 2009-2011: Email Backend
- 2011-2013: Cloud Control Panel
- 2012-2013: Team Lead

This Talk

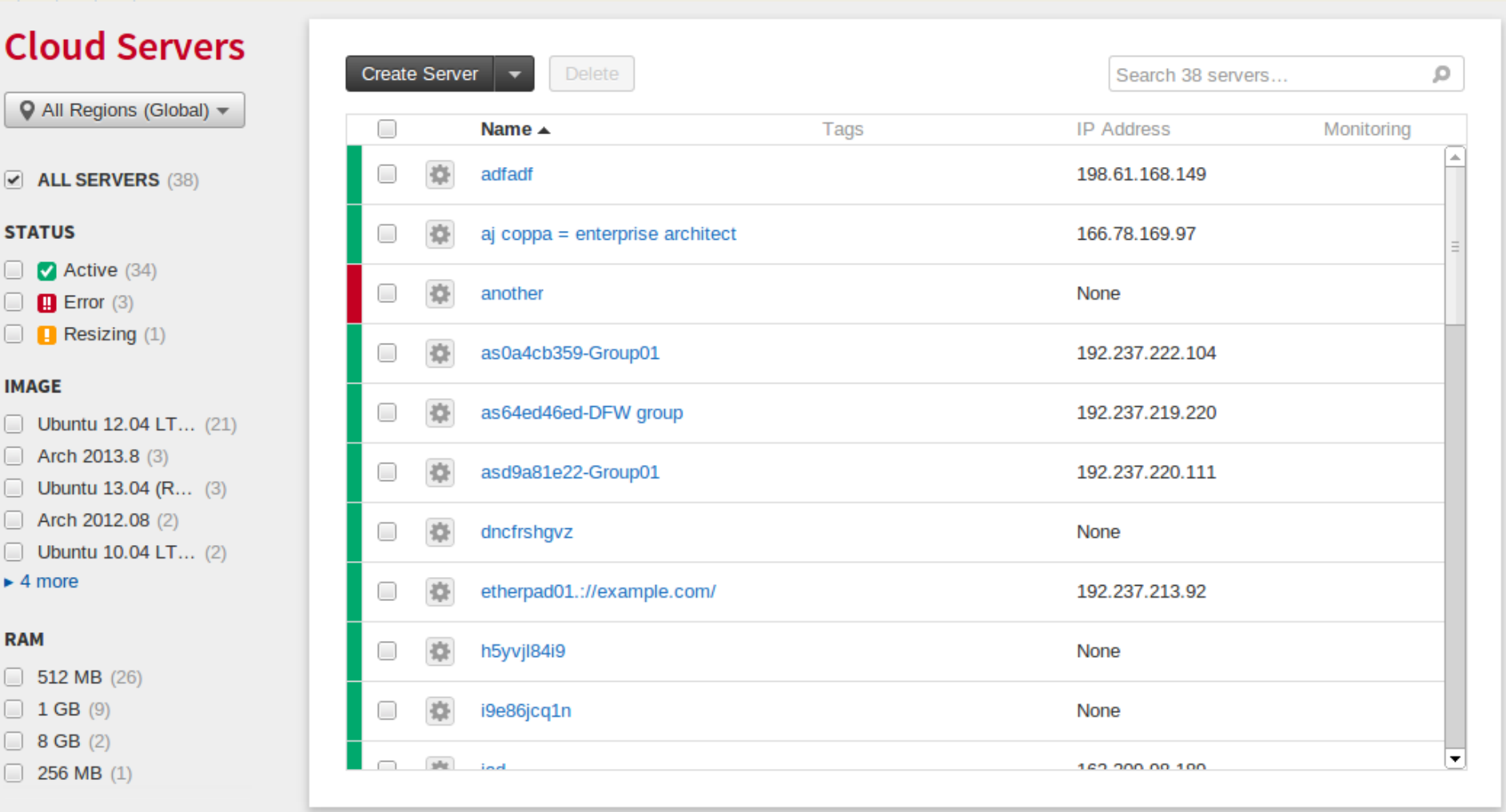
What system properties are required for development “at scale”?

How do you make these properties true?

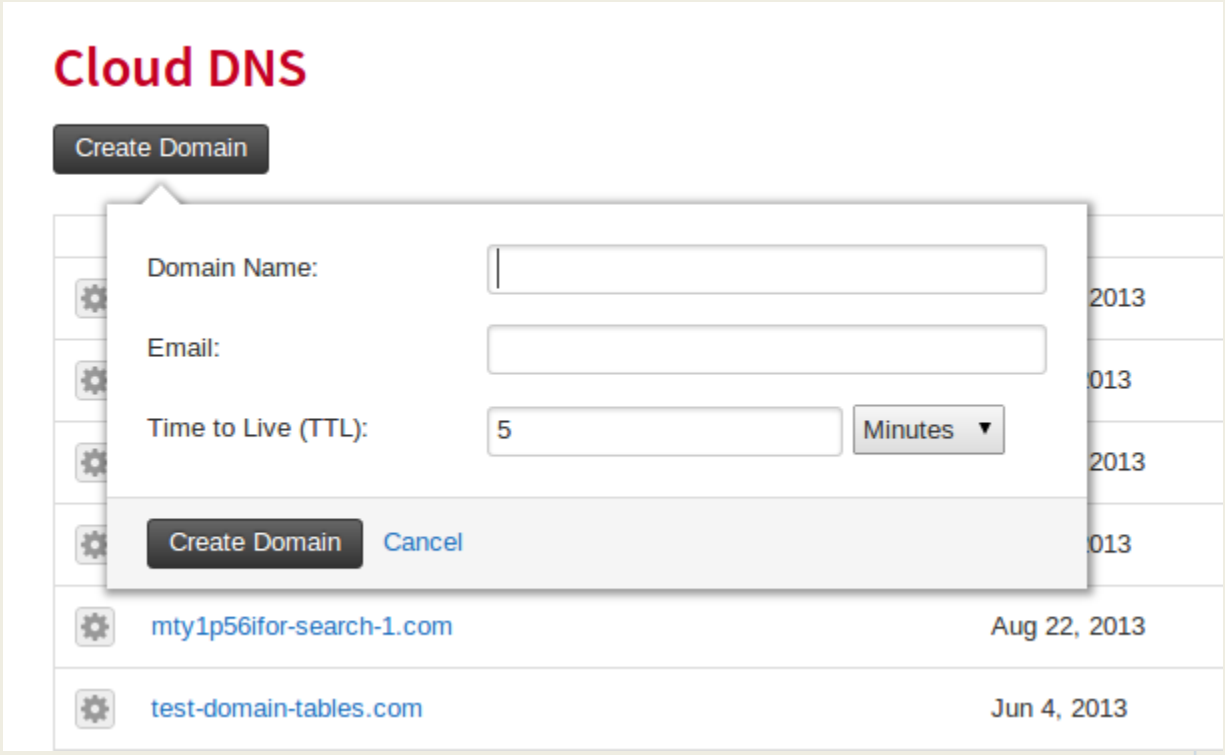
Our Product


rackspace
the open cloud company
Welcome to the Cloud Control Panel!
Username
reachnovadevops
Password
Log In
Need help logging in?
Release Notes (9/3/2013)
New to Rackspace? Join the Cloud Now »

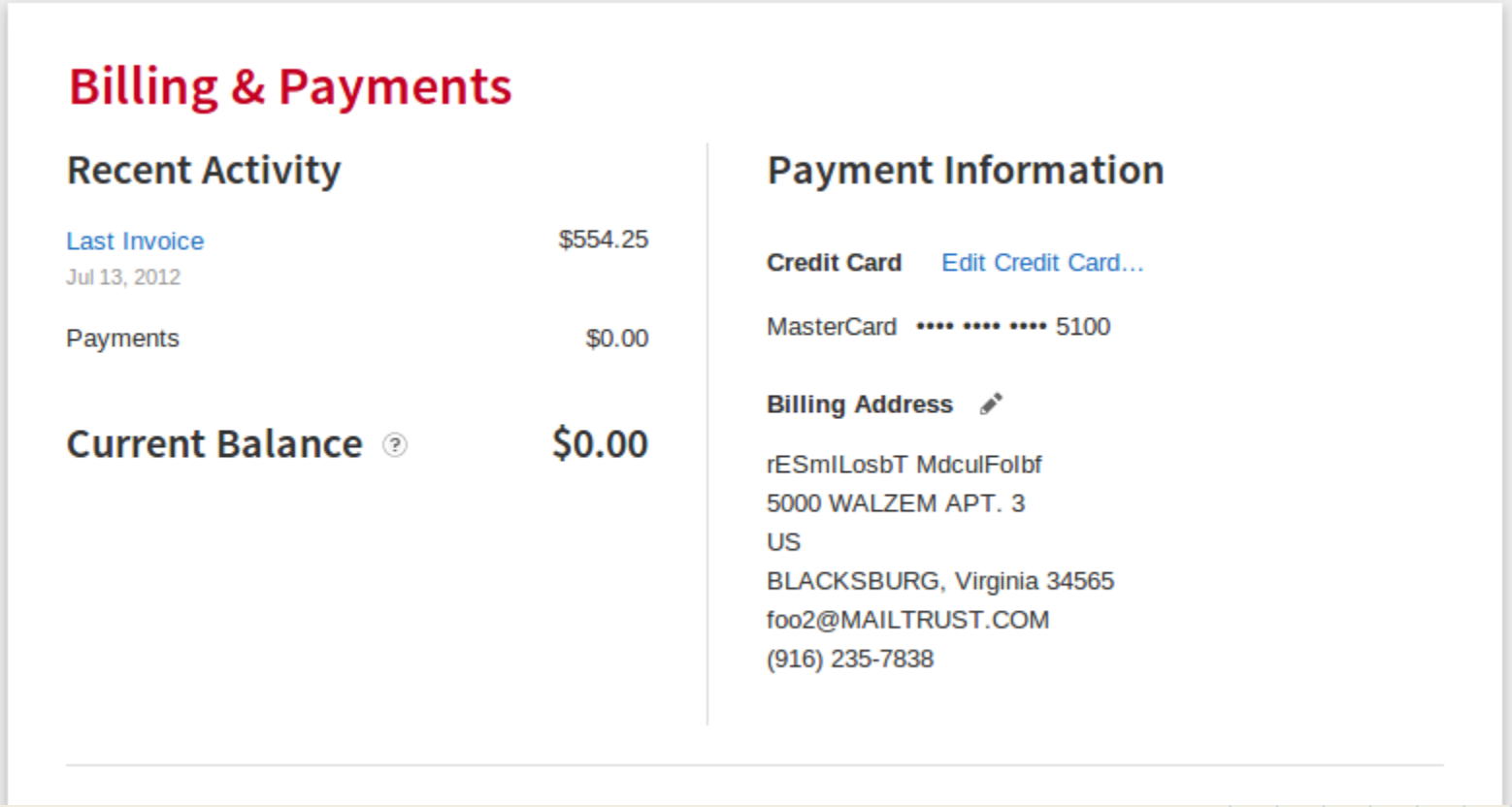
Our Product


Cloud Servers
All Regions (Global)
ALL SERVERS (38)
STATUS
Active (34)
Error (3)
Resizing (1)
IMAGE
Ubuntu 12.04 LT… (21)
Arch 2013.8 (3)
Ubuntu 13.04 (R… (3)
Arch 2012.08 (2)
Ubuntu 10.04 LT… (2)
4 more
RAM
512 MB (26)
1 GB (9)
8 GB (2)
256 MB (1)
Create Server
Delete
Search 38 servers…
Name
Tags
IP Address
Monitoring
adfadf 198.61.168.149
aj coppa = enterprise architect 166.78.169.97
another None
as0a4cb359-Group01 192.237.222.104
as64ed46ed-DFW group 192.237.219.220
asd9a81e22-Group01 192.237.220.111
dncfrshgvz None
etherpad01.://example.com/ 192.237.213.92
h5yvjl84i9 None
i9e86jcq1n None

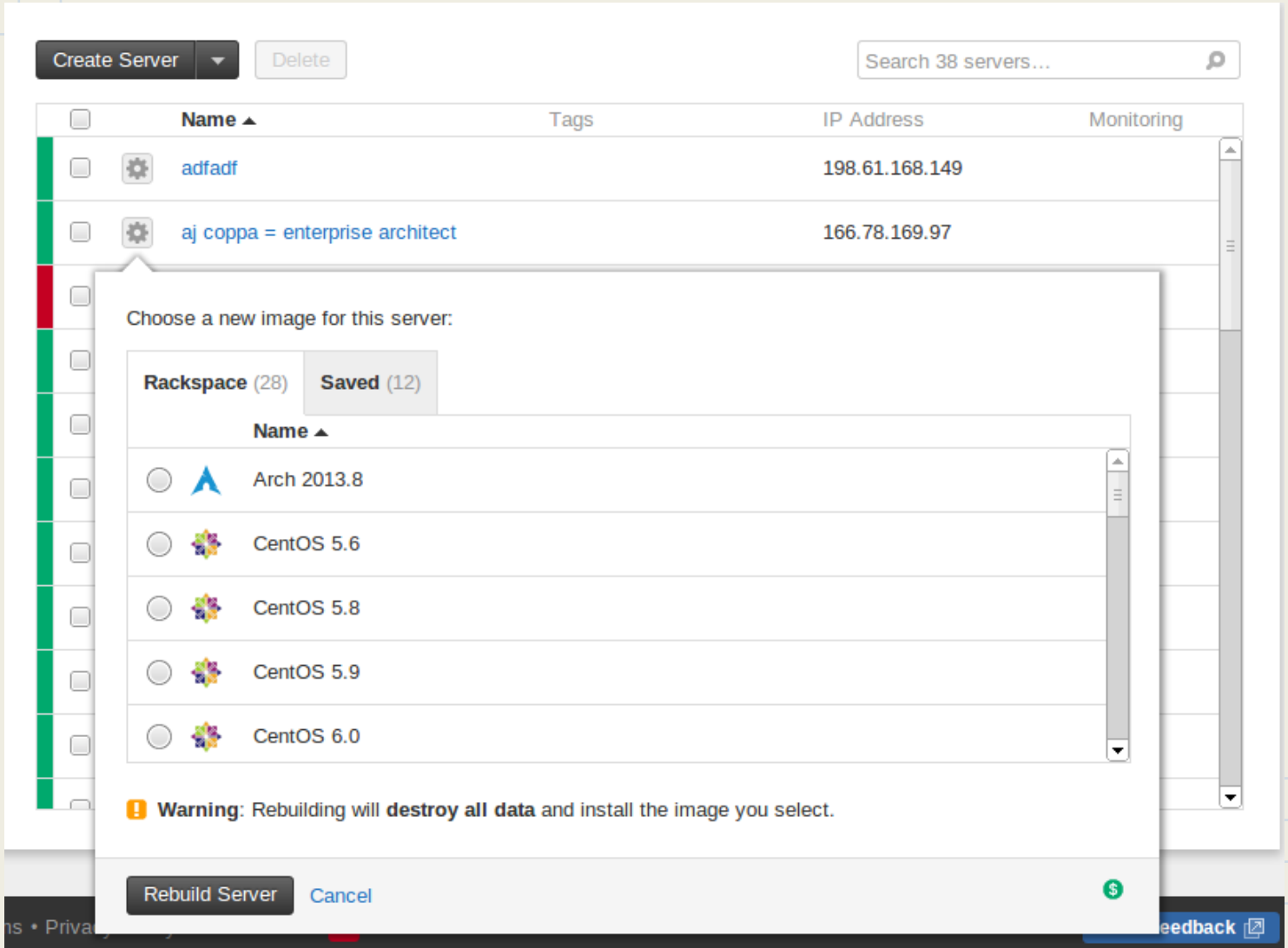
Our Product


Cloud DNS
Create Domain
Domain Name:
Email:
Time to Live (TTL):
5
Minutes
Create Domain
Cancel
2013
013
2013
013
mty1p56ifor-search-1.com
Aug 22, 2013
test-domain-tables.com
Jun 4, 2013

Our Product


Billing & Payments
Recent Activity
Last Invoice
Jul 13, 2012
$554.25
Payments
$0.00
Current Balance
$0.00
Payment Information
Credit Card
Edit Credit Card...
MasterCard •••• •••• •••• 5100
Billing Address
rESmILosbT MdculFolbf
5000 WALZEM APT. 3
US
BLACKSBURG, Virginia 34565
foo2@MAILTRUST.COM
(916) 235-7838

Our Product


Create Server
Delete
Search 38 servers…
Name
Tags
IP Address
Monitoring
adfadf
198.61.168.149
aj coppa = enterprise architect
166.78.169.97
Choose a new image for this server:
Rackspace (28)
Saved (12)
Name
Arch 2013.8
CentOS 5.6
CentOS 5.8
CentOS 5.9
CentOS 6.0
Warning: Rebuilding will destroy all data and install the image you select.
Rebuild Server
Cancel

Our Product's Tech Stack

- JavaScript
- HTML
- CSS
- Python (Django, Twisted)
- Node.js
- Ruby (browser tests)
- Linux
- Apache
- Chef
- (Probably some others I have forgotten)

Development at Scale

Multiple streams of work

200k lines of code

100+ servers

- more on this later

40+ Selenium suites

- run every time code is changed
- all finish within 15 minutes

Product Development

Product: *what should we build?*

Engineering: *how should we build it?*

Product and engineering work together!

Main Product Challenges

B2B application, ~6000 unique logins a day

Many upstream requests (80 req/second at peak)

Many feature requests:

- customers
- internal teams in Rackspace (incremental)
- leadership at Rackspace (new projects)

Engineering Concerns

Availability

Visibility

Uniformity

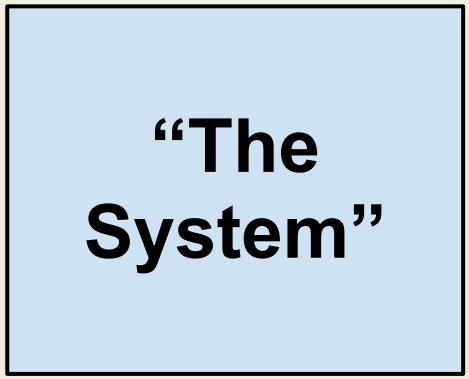
Availability

The site must ***always*** remain up

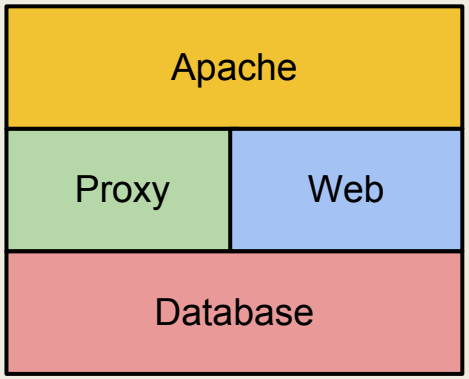
Architecting for Availability

Application made of building blocks rather than one monolithic application

Monolithic



Distributed



Availability: Single Machine

All machines in our infrastructure are part of a redundant group

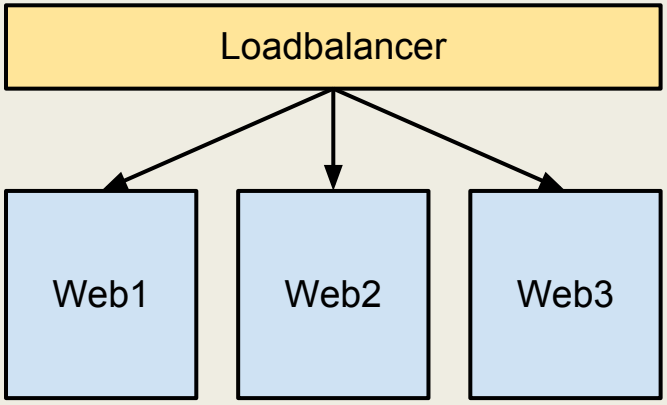
Applications hit load balancers

Failing machines do not get traffic

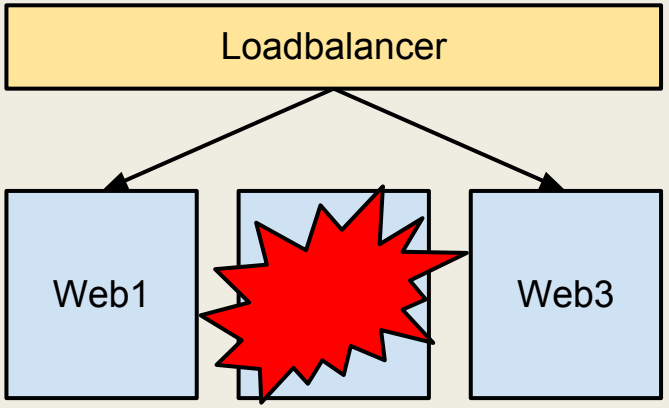
Only the database stores data

Availability: Single Machine

Normal Operation



Machine Failure



Availability: Datacenters

What if a datacenter fails?

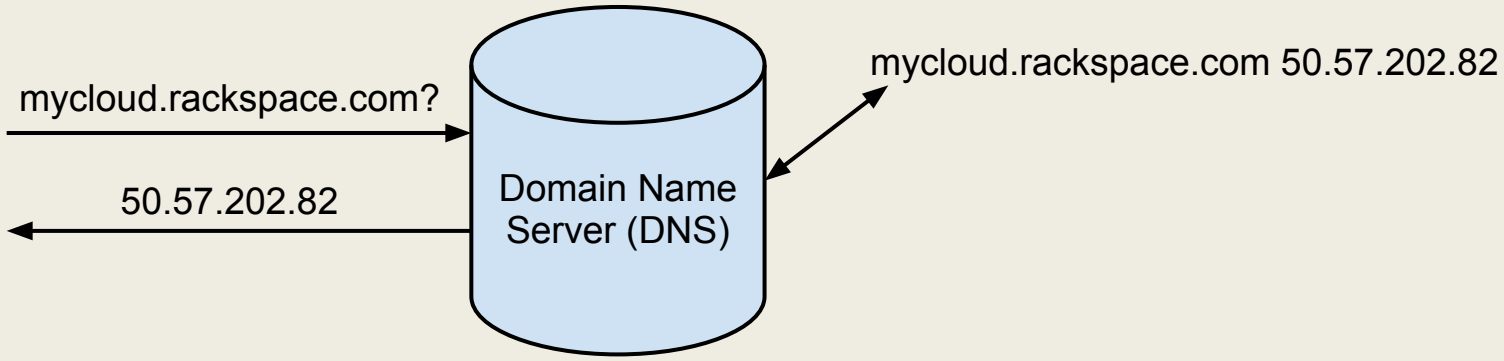
This is real:
2007: Truck hits transformer -- Rackspace Dallas goes down
2012: “derecho” hits Virginia -- AWS US-East out for days

During outages, customers still use the Control Panel!

Availability: Datacenters

How DNS works: *hostnames* and *ip addresses*

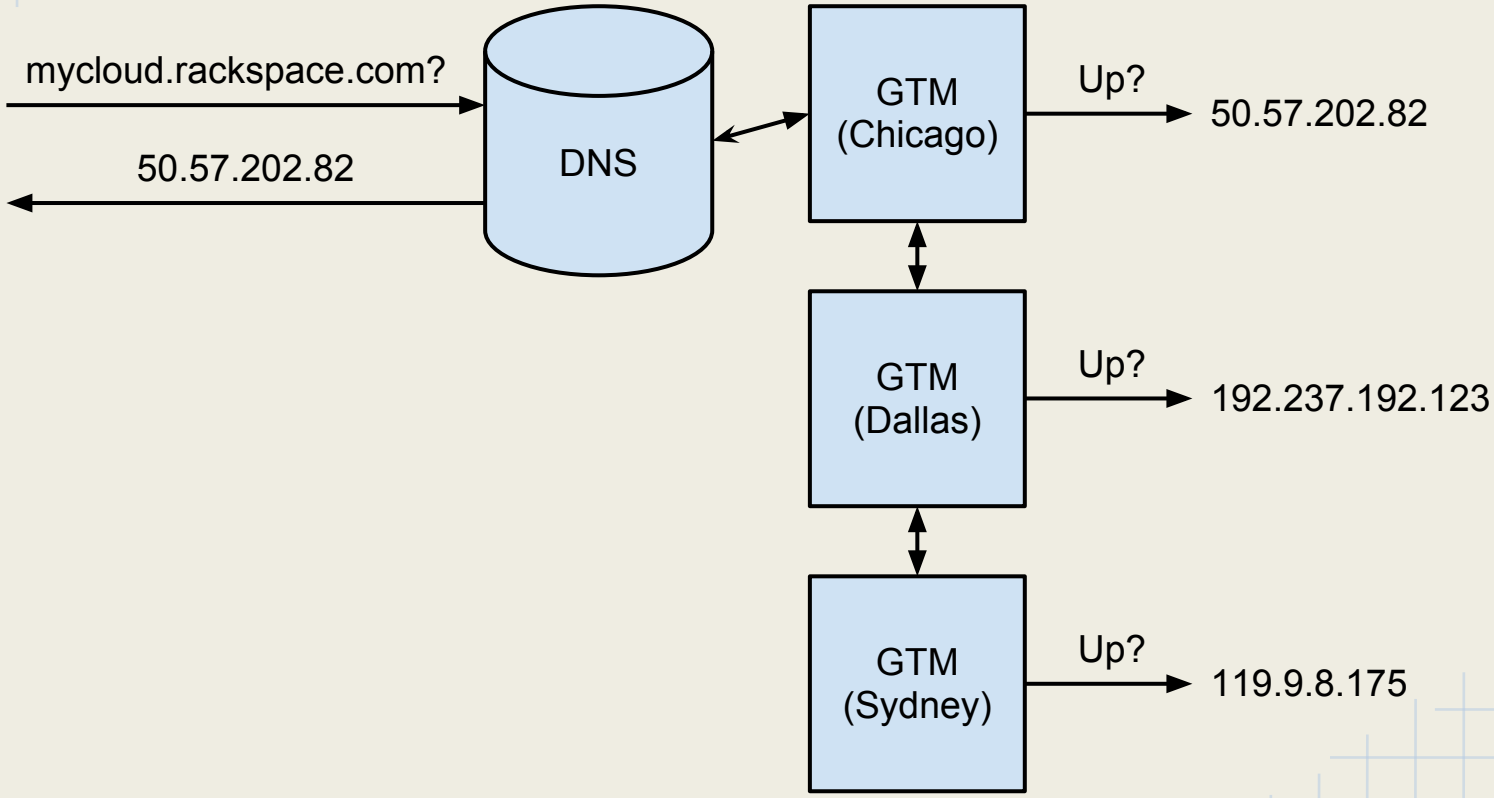
You register your site at an IP and that's that



What if 50.57.202.82 is down?

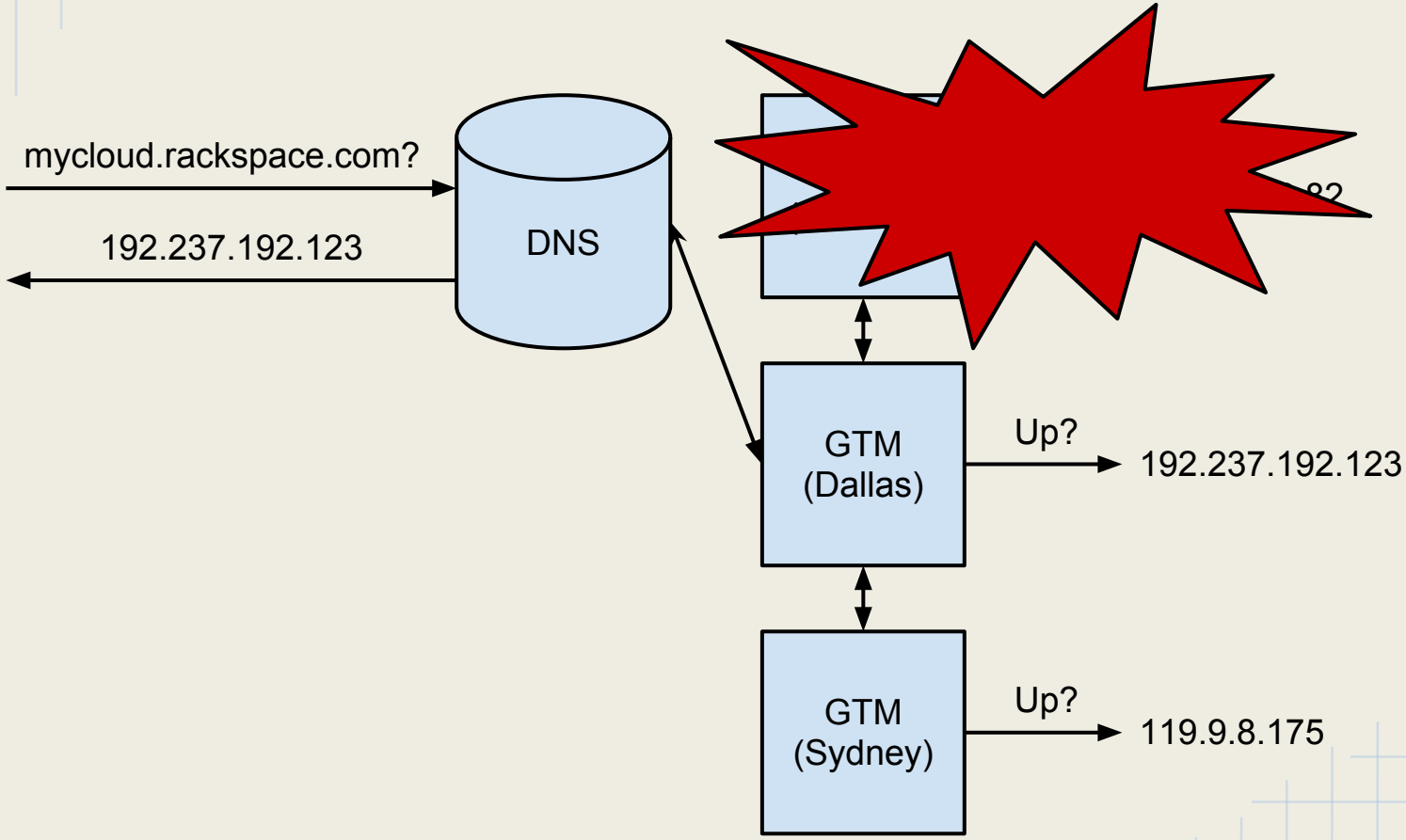
Availability: Datacenters

Support failure with *global traffic managers*



Availability: Datacenters

Datacenter failure



Availability: Datacenters

Failure of DNS handled through multiple NS records

- NS records: who owns a hostname

Having multi-datacenter redundancy also saves us from outages that are our fault

Mindset: if there is a problem you don't understand, fail the datacenter over

Availability: Deployment

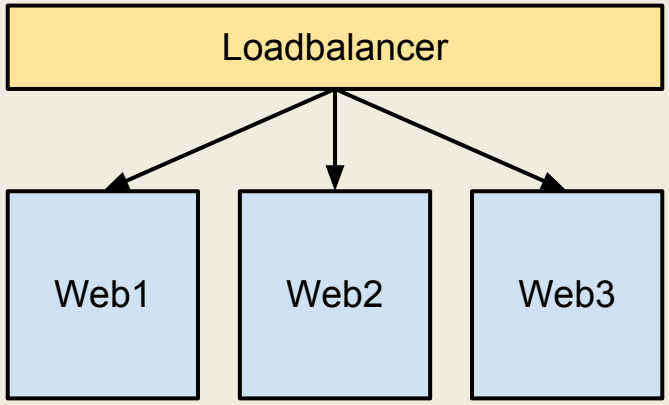
Rolling out new code needs to be zero-impact

Releasing new features:

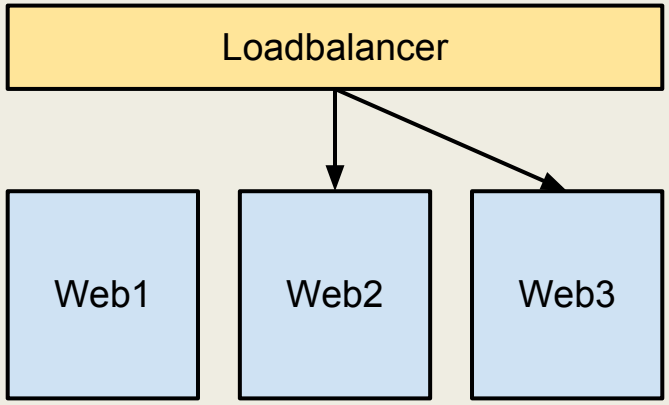
- **No downtime!**

Graceful service restarts

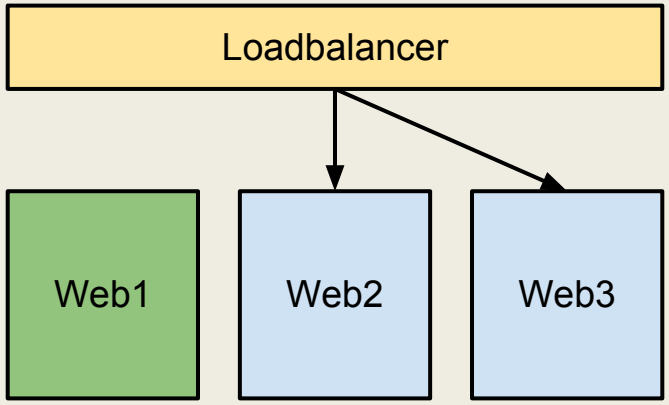
Availability: Deployment



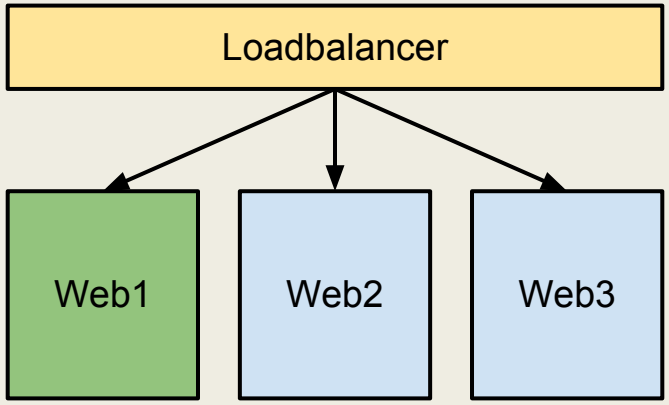
Availability: Deployment



Availability: Deployment



Availability: Deployment



Availability: Deployment

Large multi-month project: Server list rewrite

New code developed while maintaining old code

New code enabled for certain percentage of accounts

Enable new code for everyone, then delete old code

Visibility

What is happening on the site?

Need visibility to:

- diagnose problems
- get feedback on built features
- plan for the future

Visibility: Monitoring

Monitoring: scripts that check system invariants

“Is the login page available?”
“Is MySQL running?”
“Can the machine running Django connect to the database?”

Visibility: Monitoring

Remote checks: done from a remote outside your infrastructure

What the customer sees

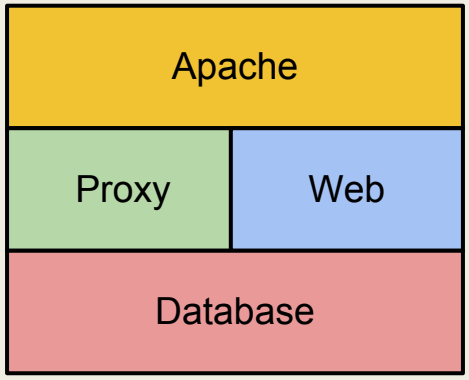
Agent checks: run on your machines

What the machine sees

Visibility: Monitoring

Example checks in our infrastructure:

- “Next Hop”: can one machine see another?
- Health check: path that returns “OK” if it can use all of the services it needs
- Infrastructure checks: Memory, CPU, Load
- Ping checks: Is a machine up?



Visibility: Monitoring

We use Rackspace Cloud Monitoring

(Compare with Nagios)

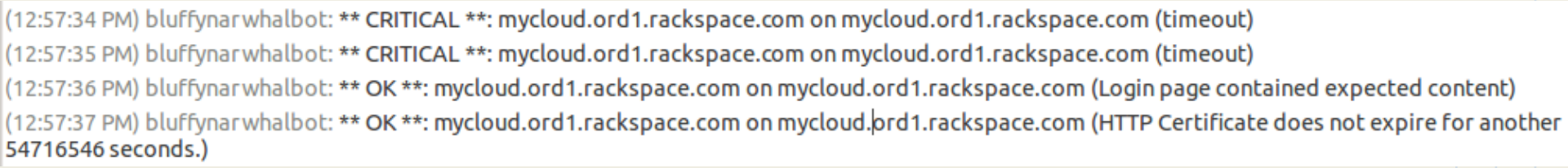
Monitoring best practice:

only alert when there is a real issue!

Visibility: Monitoring

When a check fails it notifies IRC

Some alarm flaps we have been having lately (that we are working on :))



Visibility: Analytics

We need to understand how customers are using the site

We want to integrate customer data into other sources gathered across Rackspace

Goal is to guide future product development

Visibility: Analytics

Integrate site with analytics libraries

- Google Analytics
- CoreMetrics (we use this)
- lots of others

Learn things about:

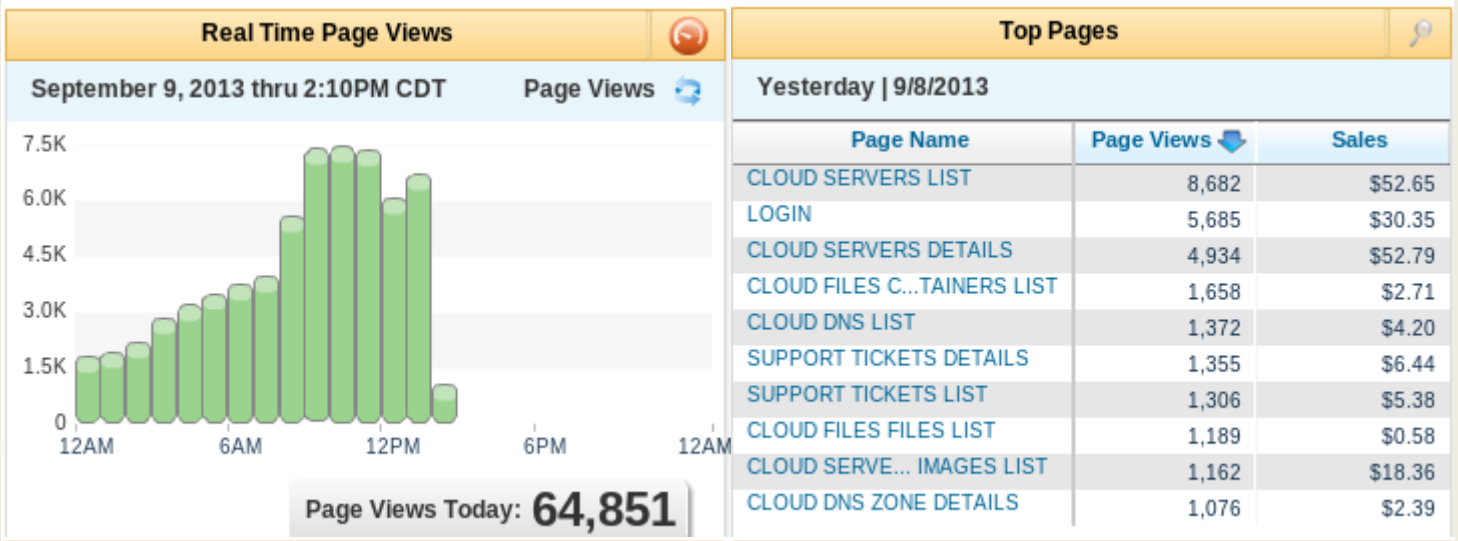
- Who is on your site (USA, China, etc)
- How long people usually stay
- How much $$$ people are spending

Visibility: Analytics

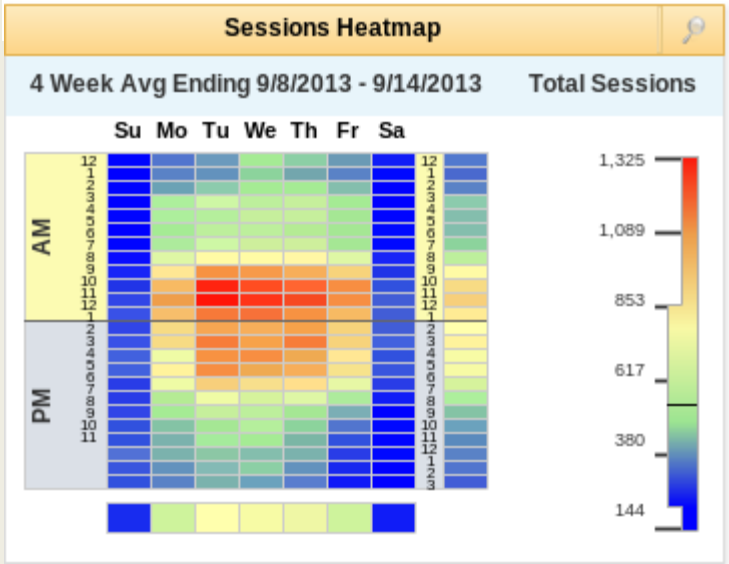
We call CoreMetrics when:

- a page loads
- a popover is displayed and ‘executed’
- you create a new server

Visibility: Analytics



Page Name	Page Views	Sales
CLOUD SERVERS LIST	8,682	$52.65
LOGIN	5,685	$30.35
CLOUD SERVERS DETAILS	4,934	$52.79
CLOUD FILES C...TAINERS LIST	1,658	$2.71
CLOUD DNS LIST	1,372	$4.20
SUPPORT TICKETS DETAILS	1,355	$6.44
SUPPORT TICKETS LIST	1,306	$5.38
CLOUD FILES FILES LIST	1,189	$0.58
CLOUD SERVE... IMAGES LIST	1,162	$18.36
CLOUD DNS ZONE DETAILS	1,076	$2.39



Visibility: Logging

Bug report: “X doesn’t work!”

We need to be able to understand problems that customers are seeing

Visibility: Logging

Server-side logging

- All requests to the server are logged
- All logs contain **username**, **level**, and a **description**
- Log levels: DEBUG, INFO, ERROR
- **Only ERROR when there is an error!**

```
2013-09-09_18:38:10.64763 INFO     User reachnovadevops,
received proxy response - GET https://storage101.dfw1.
clouddrive.com/v1/MossoCloudFS_83bdcc37-9ef7-4eee-8820-
6ac2945b7a61/?limit=100 200
```

Visibility: Logging

Our application does not often interact with the server! (single-page web application)

Client-side logging

- WebSockets: two-way communication over TCP
- socket.io: cross-browser WebSocket implementation (node.js)

Visibility: Logging

Client-Side

```
socket.emit('log', 'Created a server', {
  'username': 'Dave',
  'accountId': 324059,
  'level': 'INFO',
  // more information about the customer
});
```

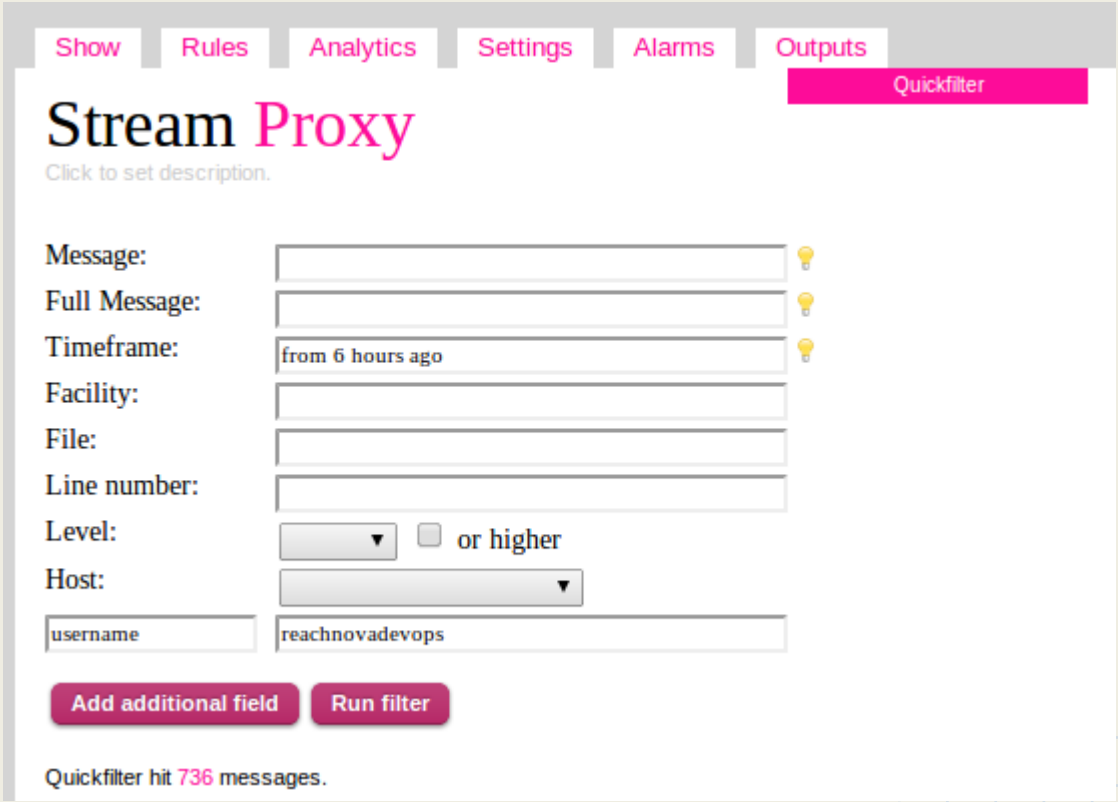
Server-Side

```
socket.on('log', function (data) {
  logLevel = data.level;
  description = data.description;
  delete data.description;
  log[logLevel](description, data);
});
```

Visibility: Logging

Filtering Logs: we use Graylog

- (see also: LogStash, Splunk, Loggly)



Visibility: Logging

Date	Host	Level	Facility	username	code	verb	Message
2013-09-10 19:47:46.668	ord1-reach-prxy0.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://monitoring.api.rackspacecloud.com/v1.0/663051/views/overview/?limit=1000 200
2013-09-10 19:47:33.156	ord1-reach-prxy0.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://syd.servers.api.rackspacecloud.com/v2/663051/images/detail 200
2013-09-10 19:47:32.458	ord1-reach-prxy1.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://ord.servers.api.rackspacecloud.com/v2/663051/images/detail 200
2013-09-10 19:47:32.189	ord1-reach-prxy1.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://iad.servers.api.rackspacecloud.com/v2/663051/images/detail 200
2013-09-10 19:47:32.125	ord1-reach-prxy2.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://dfw.servers.api.rackspacecloud.com/v2/663051/images/detail 200
2013-09-10 19:47:31.334	ord1-reach-prxy2.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://servers.api.rackspacecloud.com/v1.1/663051/servers/detail/?cache-busting=1378842448928 200
2013-09-10 19:47:31.286	ord1-reach-prxy0.k1k.me	Info	reach.twisted.proxy	reachnovadevops	200	GET	User reachnovadevops, received proxy response - GET https://servers.api.rackspacecloud.com/v1.1/663051/images/detail?cache-busting=1378842448935 200

Visibility

We need to understand what load the site has

- Requests per second
- CPU/Memory/Network usage
- Interesting things that happen (logins)

Visibility: Statistics

Statistics:

- Counters - when an event happens (e.g. login)
- Timers - when something takes time (e.g. upstream request)
- Gauges - when something is at a certain metrics (e.g. CPU load, 95%)

Etsy: Measure Anything, Measure Everything
We use Etsy's stats daemon (`statsd`)

Visibility: Statistics

statsd receives stats over UDP

Every minute, statsd flushes stats to graphite

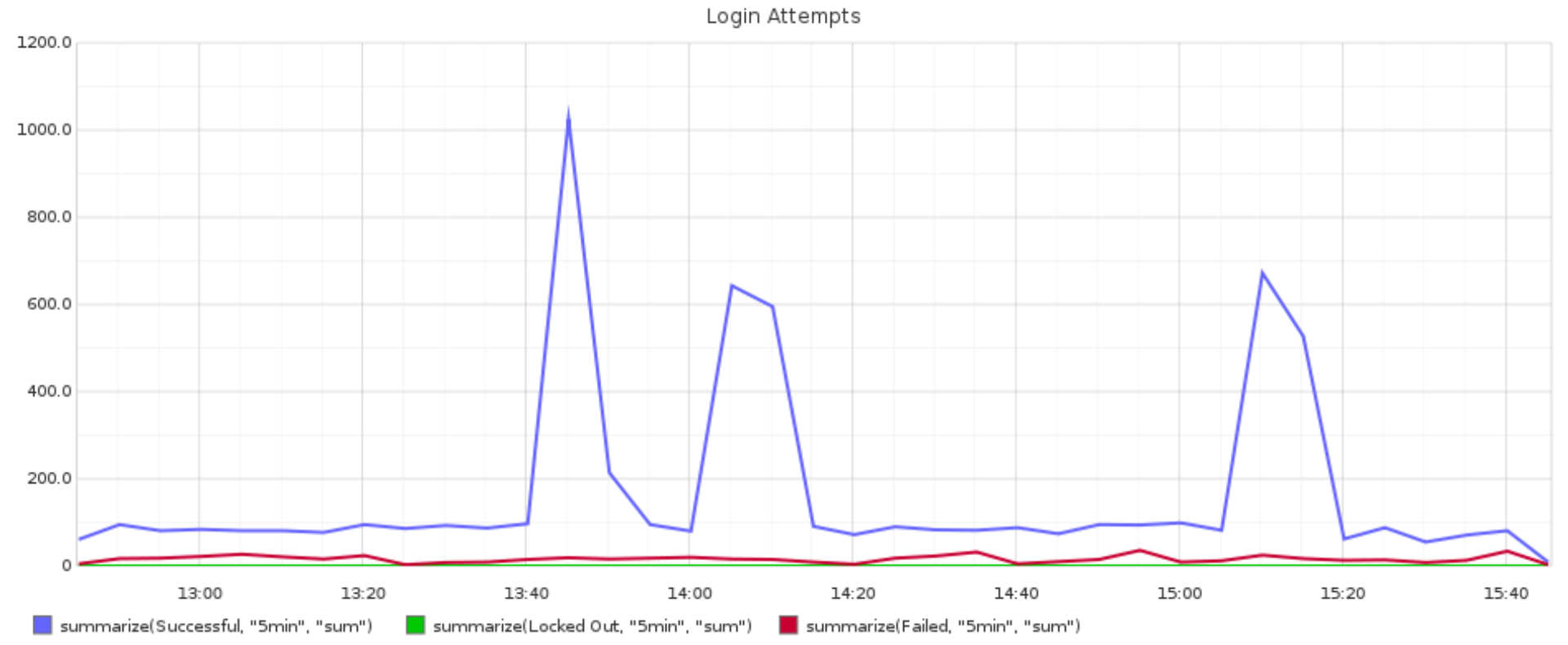
Graphite: real-time graph visualization

- send stats to graphite
- graph stats over time period with parameters

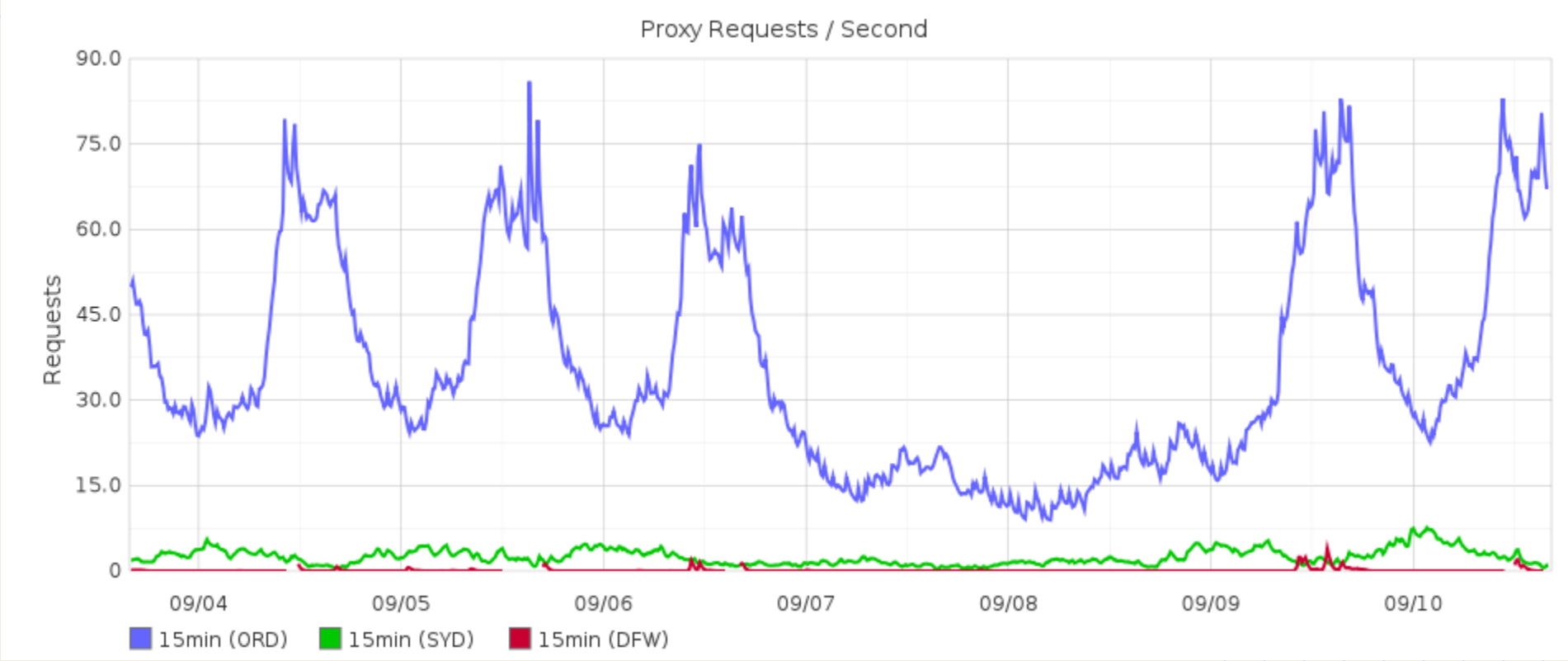
Visibility: Statistics

```
if locked_out:
    stats_client.incr('auth.login.lockout')
else:
    if not login_success:
        stats_client.incr('auth.login.failure')
    else:
        stats_client.incr('auth.login.success')
```

Visibility: Statistics



Visibility: Statistics



Uniformity

No surprises on our servers!

Every server of a certain type needs to behave exactly the same

If a server is misbehaving (for whatever reason) it should be able to be deleted and recreated

Uniformity: Cloud Servers

Infrastructure runs on Rackspace Cloud Servers

“Cloud Servers”

- ‘slices’ of compute instances
- Most of our infrastructure runs on 8 GB Next Generation nodes (OpenStack-based)
- each slice is the same: no differences in hardware, no differences in data drives, etc

Uniformity: Chef

Configuration management: set of scripts that install application, setup config files, install necessary libraries

Idempotent: multiple runs have no result

We use *Chef* for configuration management

Uniformity: Chef

```
web_app "reach-lb" do
  template "reach-lb.conf.erb"
  server_name node['hostname']
  server_aliases node['reach']['aliases']
  log_dir node['apache']['log_dir']
  web_nodes web_nodes
  listen_ports node[:apache][:listen_ports]

  asset_paths = ['/data/ck/cloudkick/webapp/site_media']
  asset_paths << '/data/releases/shared/assets'
  asset_directories asset_paths
end
```

Uniformity: Preproduction

Preproduction: a lower environment that mimics production

Settings exactly the same

- Preprod features are production features
- Slightly fewer machines

Changes that go to production are **always** validated in preproduction.

Where We're Going

Availability

- Engineer data store for high availability

Visibility

- Aggregate all logs in all datacenters to one place
- Per-customer statistics

Uniformity

- Keep on being uniform :)

Postscript: My Day-to-Day

Software development is about *communication!*

Solving hard problems together as a team