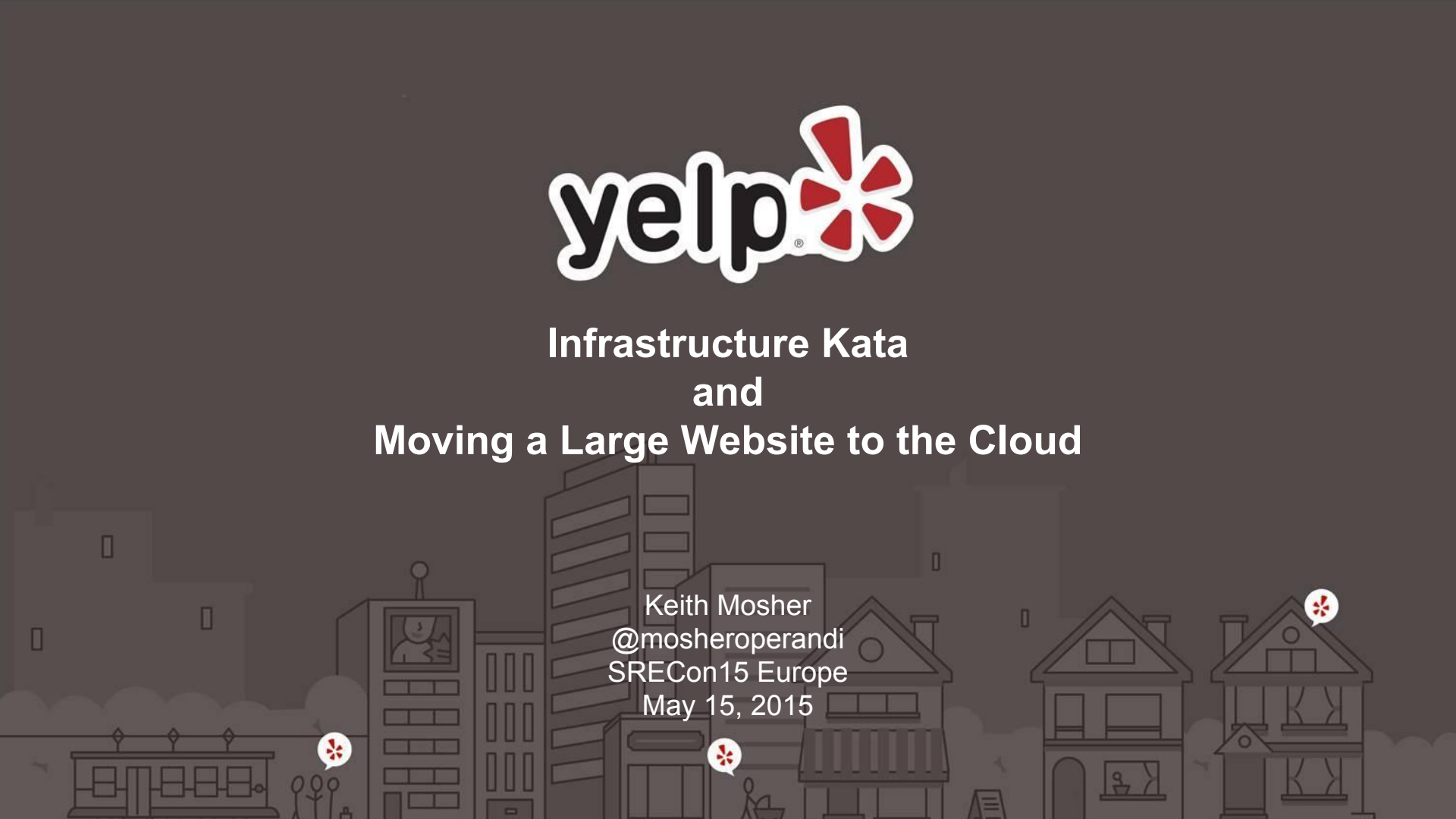
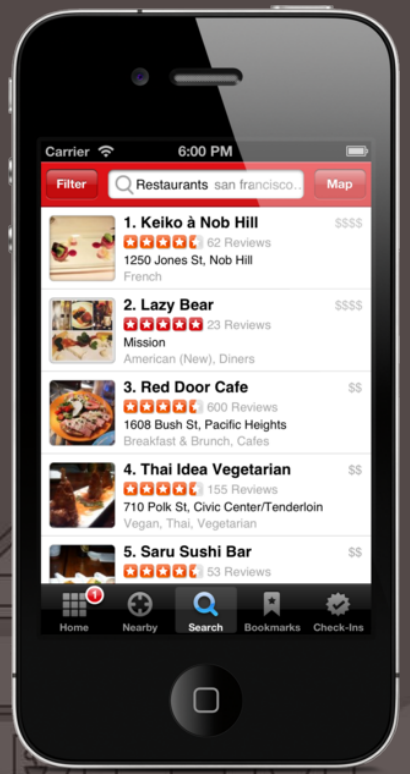

yelp
Infrastructure Kata
and
Moving a Large Website to the Cloud
Keith Mosher
@mosheroperandi
SRECon15 Europe
May 15, 2015

yelp

- ★ Website/app for finding and reviewing local businesses
- ★ Founded in 2004
- ★ 135 Million monthly unique users
- ★ 77 Million review
- ★ Available in 31 countries



Yelp Servers circa 2013

Everything is bear metal

Yelp Operations circa 2013

- ★ Hardware is hard
- ★ Processes are slow:
  - ○ Provisioning a new machine: hours (or sometimes days)
  - ○ Ordering new hardware: weeks
  - ○ Bring up a new datacenter: a year?
- ★ Lots of new development is bottlenecked on hardware


Solution: To the Cloud!

Land of milk, honey, and on-demand instances

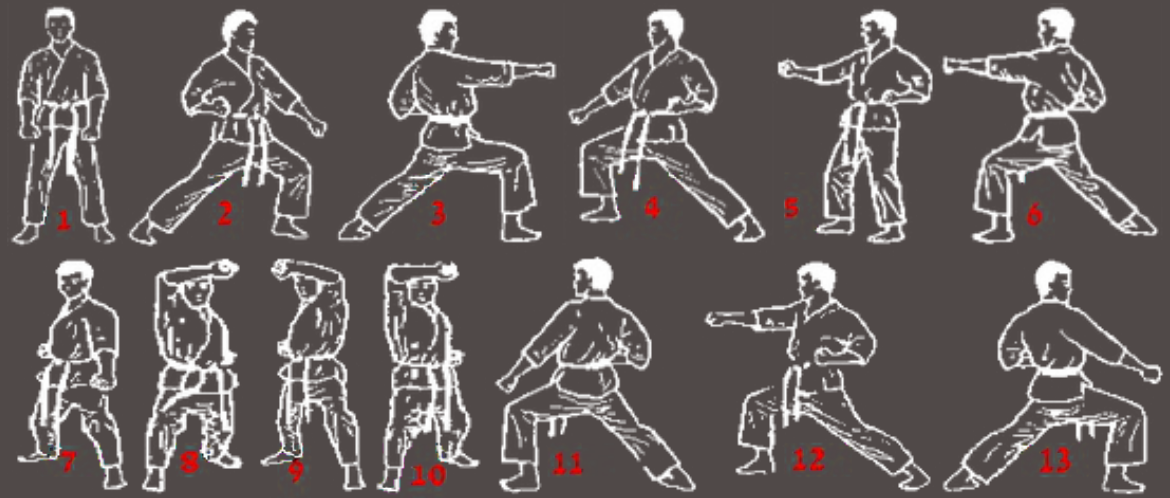
https://www.flickr.com/photos/flahertyb/7937851046/

SPOILERS: Cloud achieved

- ★ 3x as many EC2 as physical
- ★ 30% of traffic served from EC2
- ★ Fully provisioned machines are available within minutes
- ★ Done without sacrificing architectural coherency


ka·ta (形, *lit.* "form") *n.*

Structured routines designed to hone a skill through practice and repetition.



in·fra·struc·ture ka·ta

An operation, that let's face it, you'll be doing a lot of, so while you're busy repeating it, you might as well treat it as practice.

Mastery is measured not at the individual level, but at the organizational and infrastructural level.

Before You Can Start

- ★ You actually need a procedure
- ★ Do the work (probably terribly manual)
- ★ **Write down everything you do**
- ★ **EVERYTHING**
- ★ Still, something > nothing


Practice the Kata

- ★ Work through the procedure
- ★ Identify weak points. Address the worst.
- ★ The best way to practice, is to have someone else practice
- ★ Eventually your kata involves a single command


Mastery

- ★ Automating something doesn't make you a master
- ★ Infra changes. Bits rot. Edge cases are infrequent.
- ★ You need to keep practicing
- ★ Running your tool dozens of times a day in production is the true hallmark of mastery


The Story of AWS at Yelp

AWS @ Yelp: Services

- ★ First major use of EC2 at Yelp
- ★ Packer used to bake AMIs
- ★ Netflix Asgard used to manage deployments
- ★ Allowed launch of new ads framework

Kata

Build a Machine Image

Kata: Build a Machine Image

- ★ An important skill to keep honed
- ★ Packer makes this easy to automate
- ★ Mastery is when your CD pipeline does this for you

AWS @ Yelp: Full Stack

- ★ Backend services were just the start
- ★ Asgard didn't mesh well with the our other systems
- ★ We wanted AWS and datacenter servers to look the same
- ★ New Goal: An independent production stack in EC2


Kata

Set Up Your Entire Stack

Kata

Set Up a Web Server

Starting Small

- ★ We got a single web server running on EC2
- ★ All supporting infrastructure sourced from nearby datacenter

Starting Small

- ★ Even getting this far involves two crucial procedures
  - Launch an instance
  - Provision an instance


Kata

Launch an Instance


Kata: Launch an Instance

★ At worst:

```
aws --profile prod ec2 run-instances --
instance-type c3.4xlarge
--image-id ami-f76e8ab3
--subnet-id subnet-db2dd0b3
--user-data run-puppet.sh
--block-device-mappings (...)
```

Kata: Launch an Instance

- ★ Better:

```
clops launch --instance-type c3.4xlarge

web15-uswest1aprod.prod.yelpcorp.com
```

- ★ Looks up values
- ★ Enforces tagging conventions


Kata

Provision an Instance

Kata: Provision an Instance

- ★ This should already be a well developed procedure
- ★ Might lose some benefits of your imager
- ★ Our base AMI comes ready to run Puppet

Kata

Set Up a Database Server

Kata

Set Up a Load Balancer


Kata

Set Up a Search Server

Incremental Infrastructure

- ★ One host class at a time
- ★ Each of these can be its own kata
- ★ Don't forget to write everything down
- ★ Iterate until independent


New Pain Points

- ★ Getting a working server is quick and simple
- ★ Putting it into production configs is tedious

Kata

Put an Instance into Production

Kata: Put an Instance into Production

- ★ Need to configure:
  - ○ monitoring
  - ○ load balancers
  - ○ etc.
- ★ In 2013, this was all done by hand

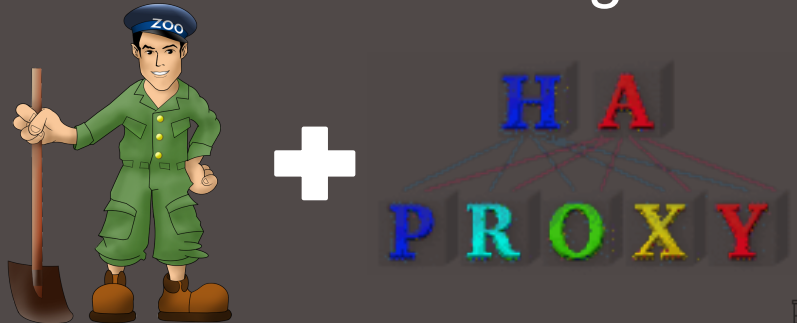
Monitoring

- ★ Nagios is not built for dynamic configuration
- ★ So we moved to Sensu (configured via Puppet)

```
monitoring_check { 'mem_free':
    team         => $team,
    page         => true,
    check_every => '1m',
    alert_after => '5m',
    runbook      => 'y/rb-generic-monitoring',
    command      => "check_mem_free -c ${crit_fraction}"
}
```

Load Balancing

- ★ Load balancer config was generated from script
- ★ Required edits and then manual applications
- ★ We switched to using AirBnB's SmartStack
  - ○ First for internal load balancing
  - ○ Later for external load balancing as well



Seagull: The Real Success Story

- ★ Wrote a new test framework (Seagull) using Marathon
- ★ Runs entirely in EC2
- ★ Automatically scales to meet demand
- ★ Cut test times in half
- ★ Coming Soon™

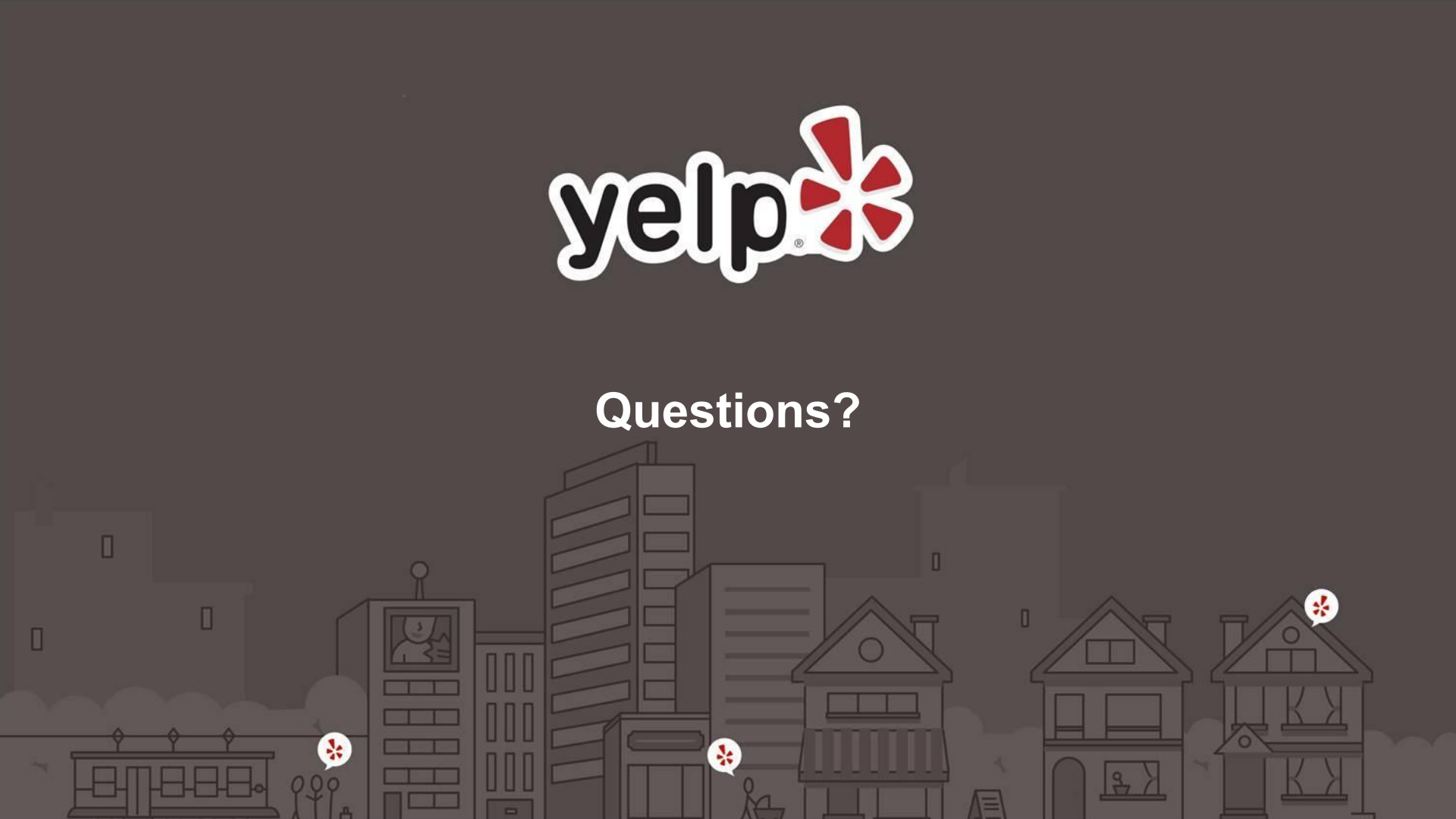
The Future

- ★ Terraform is awesome
- ★ Writing things down in code is the best way to write things down
- ★ Maybe one day we will master building environments

Conclusions

- ★ Get everyone on the same page
- ★ Treat each repetition as a practice opportunity
- ★ You don't need a foreign buzzword to understand this
- ★ Don't despair if you're stuck with a clunky application

WRITE EVERYTHING DOWN


yelp
Questions?

Links to Toys

- ★ Packer: www.packer.io/
- ★ Terraform: www.terraform.io/
- ★ Sensu: sensuapp.org/
- ★ monitoring_check: github.com/Yelp/puppet-monitoring_check/
- ★ Mesos: mesos.apache.org/
- ★ Marathon: mesosphere.github.io/marathon/
- ★ SmartStack: nerds.airbnb.com/smartstack-service-discovery-cloud/
- ★ Seagull: Watch engineeringblog.yelp.com/ for updates