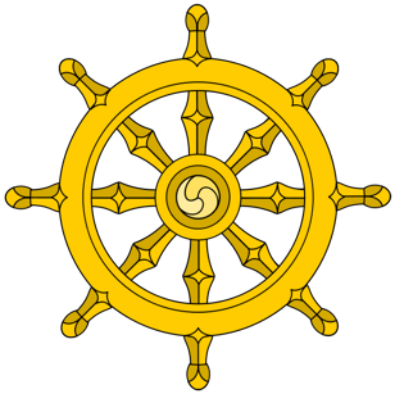


Cloud Developer's DHARMA…
redefining 'done' for Cloud applications

Daniel Bryant
Principal Consultant, Open Credo

daniel.bryant@opencredo.com
@danielbryantuk

OpenCredo

What to expect

- Problems when developing for the Cloud
  - "Lift and shift"
  - Smashing the monolith
  - Greenfield
- Some suggestions on where to focus efforts
- Tools and techniques
- Lots of information… (slides will be available)

Who Am I?

- Principal Consultant at Open Credo
  - Agile transformations
  - DevOps
  - Microservices and Cloud
- LJC Associate
- Adopt OpenJDK and JSR

Core Changes…

- Service-Oriented Architecture
- Cloud-based deployments
- DevOps Culture

Core Changes…

- Service-Oriented Architecture
  - Twitter's Story (bit.ly/1j1WbmI)
- **Cloud-based deployments**
  - **Today!**
- DevOps Culture
  - Devoxx UK talk (bit.ly/1BylnZb)
  - Previous LJC Event (bit.ly/1elVPJz)

Common Cloud Problems

TL;DR…


Not respecting the underlying environment
TOTAL
Red Bull
TOTAL

Lack of application/platform monitoring…

Bizarre failure modes…

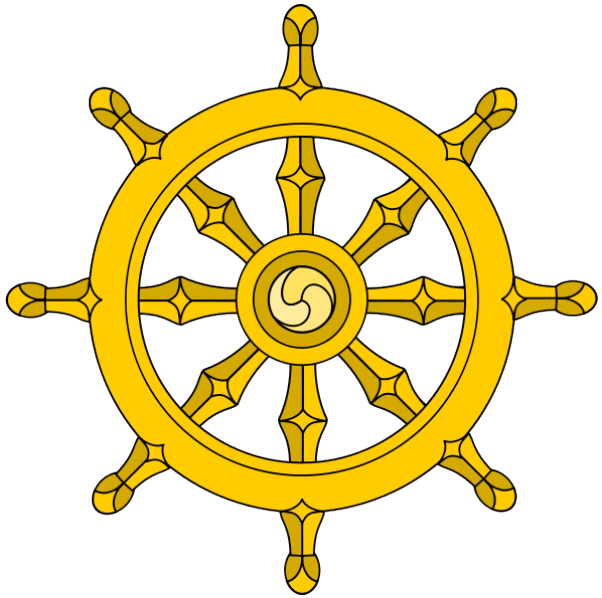

Difficult to understand
the architecture
GOOD LUCK
45
M.P.H.


Not testing in the Cloud…
(hint: here be dragons!)
Worth1000.com

We've created a "Cloud Developer's DHARMA" to act as a checklist when building Cloud apps

dharma
/ˈdɑːmə,ˈdəːmə/

noun



1. Signifies behaviors that are considered to be in accord with order that makes life and universe possible (Hinduism)
2. "cosmic law and order”, but is also applied to the teachings of the Buddha (Buddhism)

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

Resource aware

Monitored thoroughly

Antifragile

Documented (just enough)

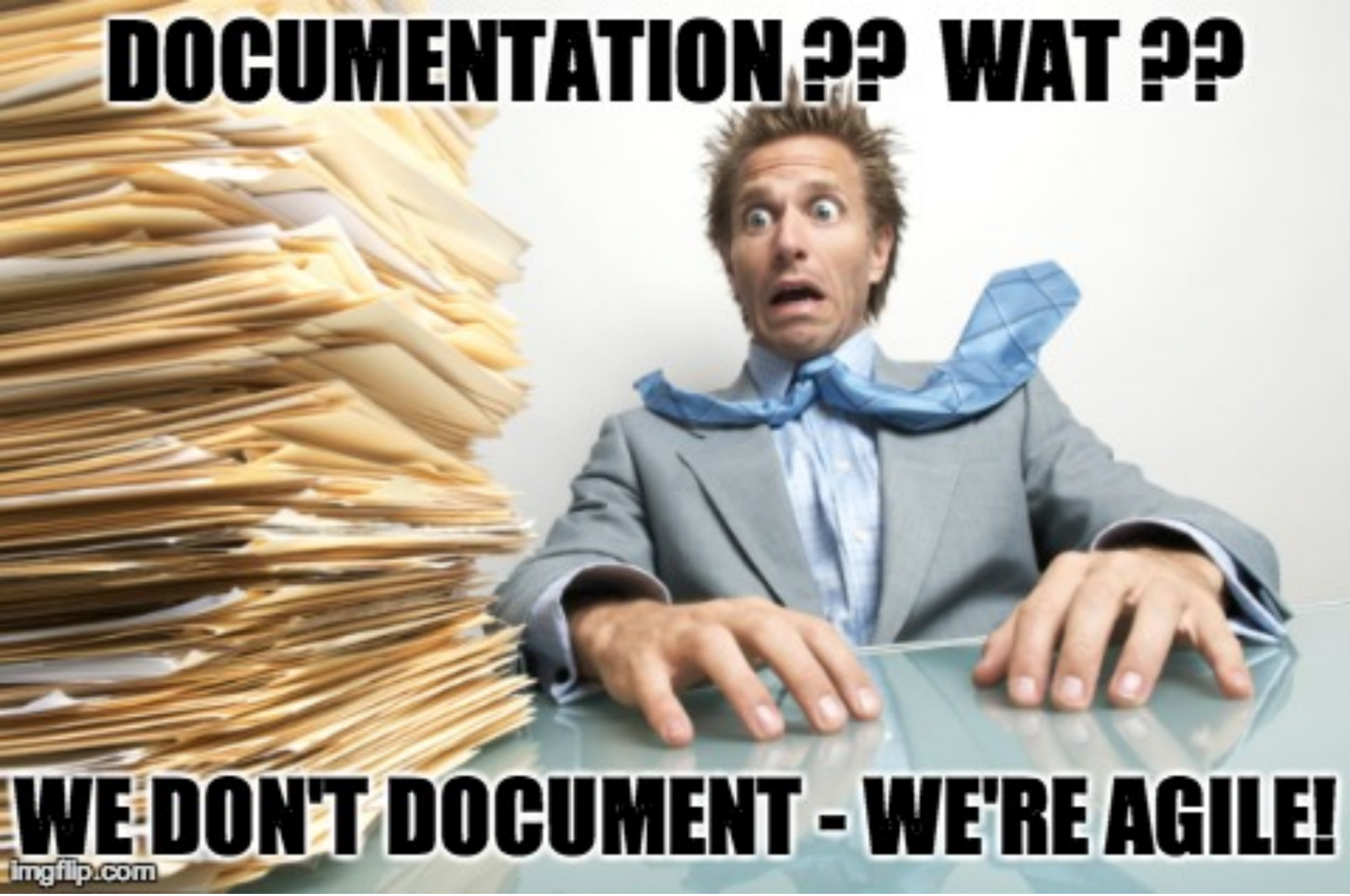
Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

Resource aware

Monitored thoroughly

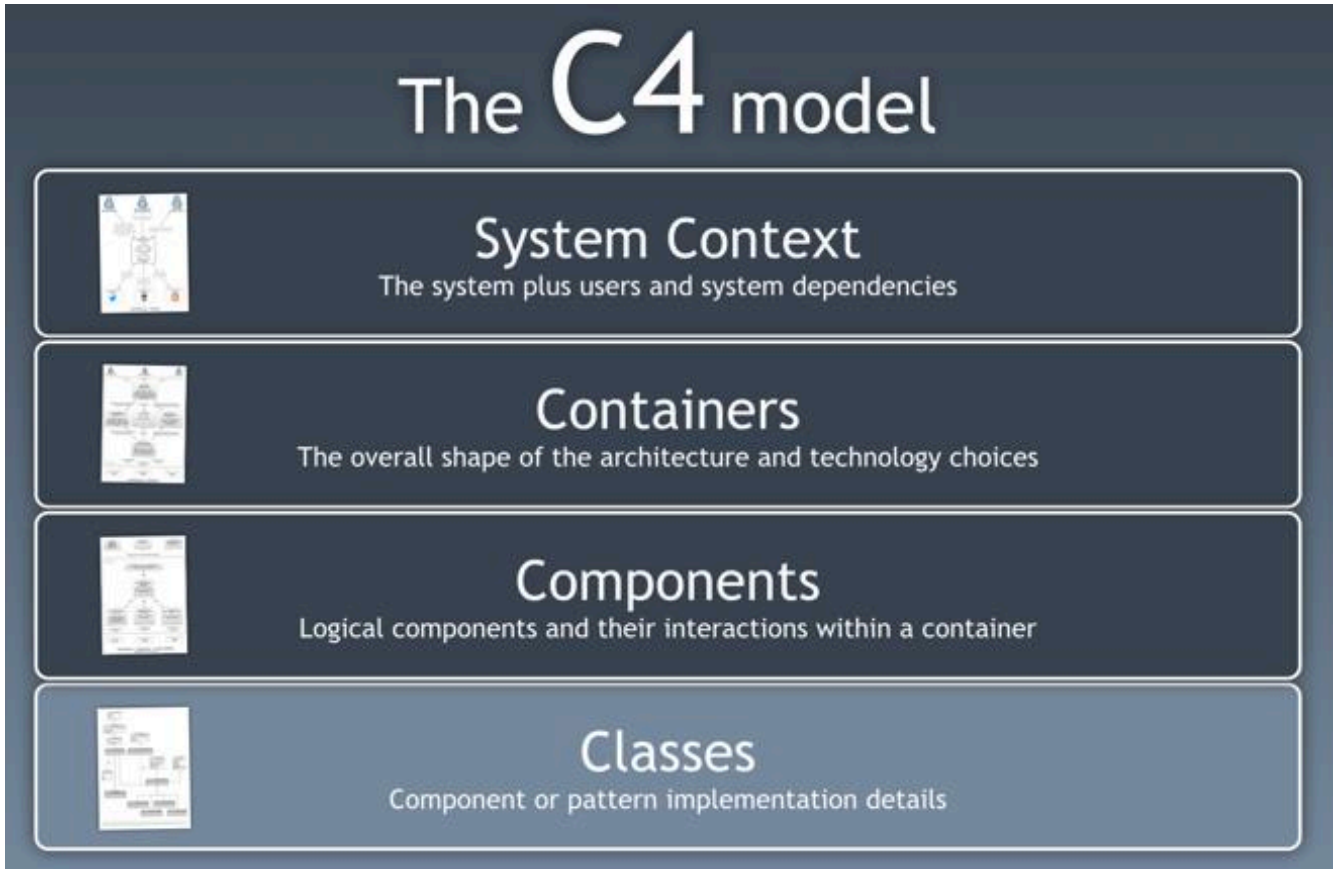
Antifragile


DOCUMENTATION ?? WAT ??
WE DON'T DOCUMENT - WE'RE AGILE!
imgflip.com

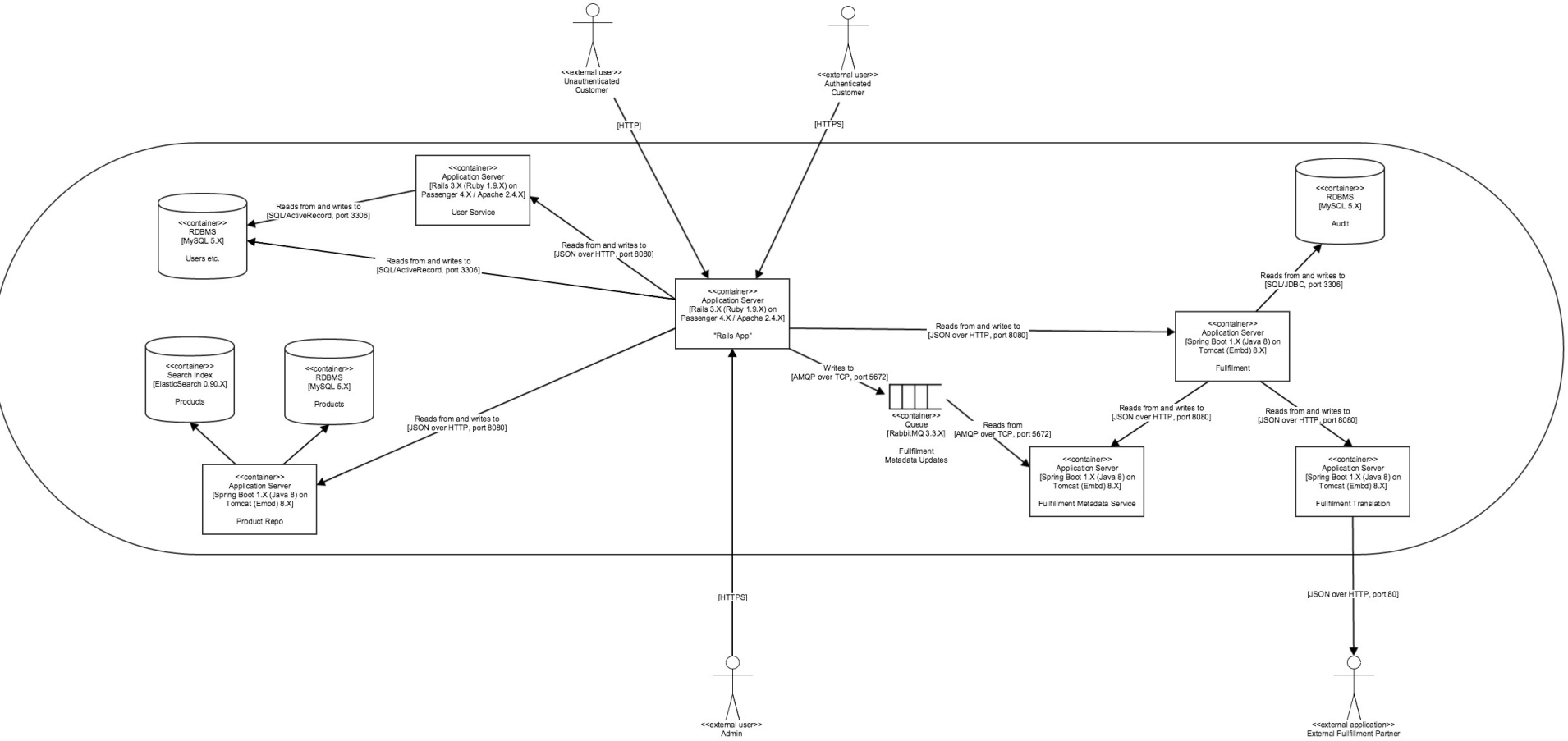
Documentation (just enough)

- Provide a map for developers
- Component purpose (and contract)
- Initialisation instructions
- Highlights areas of operational risk

Simon Brown's C4 Model



http://www.codingthearchitecture.com/


<<external user>>
Unauthenticated
Customer
<<external user>>
Authenticated
Customer
[HTTP]
[HTTPS]
<<container>>
Application Server
[Rails 3.X (Ruby 1.9.X) on
Passenger 4.X / Apache 2.4.X]
User Service
<<container>>
RDBMS
[MySQL 5.X]
Users etc.
Reads from and writes to
[SQL/ActiveRecord, port 3306]
Reads from and writes to
[JSON over HTTP, port 8080]
Reads from and writes to
[SQL/ActiveRecord, port 3306]
<<container>>
Application Server
[Rails 3.X (Ruby 1.9.X) on
Passenger 4.X / Apache 2.4.X]
"Rails App"
<<container>>
RDBMS
[MySQL 5.X]
Audit
Reads from and writes to
[SQL/JDBC, port 3306]
Reads from and writes to
[JSON over HTTP, port 8080]
<<container>>
Application Server
[Spring Boot 1.X (Java 8) on
Tomcat (Embd) 8.X]
Fulfilment
<<container>>
Search Index
[ElasticSearch 0.90.X]
Products
<<container>>
RDBMS
[MySQL 5.X]
Products
Reads from and writes to
[JSON over HTTP, port 8080]
Writes to
[AMQP over TCP, port 5672]
<<container>>
Queue
[RabbitMQ 3.3.X]
Fulifilment
Metadata Updates
Reads from
[AMQP over TCP, port 5672]
Reads from and writes to
[JSON over HTTP, port 8080]
Reads from and writes to
[JSON over HTTP, port 8080]
<<container>>
Application Server
[Spring Boot 1.X (Java 8) on
Tomcat (Embd) 8.X]
Product Repo
<<container>>
Application Server
[Spring Boot 1.X (Java 8) on
Tomcat (Embd) 8.X]
Fullfilment Metadata Service
<<container>>
Application Server
[Spring Boot 1.X (Java 8) on
Tomcat (Embd) 8.X]
Fullfilment Translation
[HTTPS]
[JSON over HTTP, port 80]
<<external user>>
Admin
<<external application>>
External Fulifillment Partner

OpenCredo

Apps Weather Regulars Calendar feedly IAT Admin IAT Stack Nagios Local Tech OpenJDK To Read Current Awesomeness » Other Bookmarks

admin-portal

Responsibilities:

Provides admin-portal UI, allowing the creation and management of various entities, such as Feed, Vouchers, Network, Brands and Merchants.

Exposes above data to internal IAT components via REST-like API

Component initialisation instructions:

Run via embedded Jetty within Fat JAR:

```
java -jar admin-ui/target/admin-ui-1.15.1-jar-with-dependencies.jar
```

Visit: localhost:8910

OR use Grunt (http://gruntjs.com/)

Install all dependencies from package.json on first checkout:

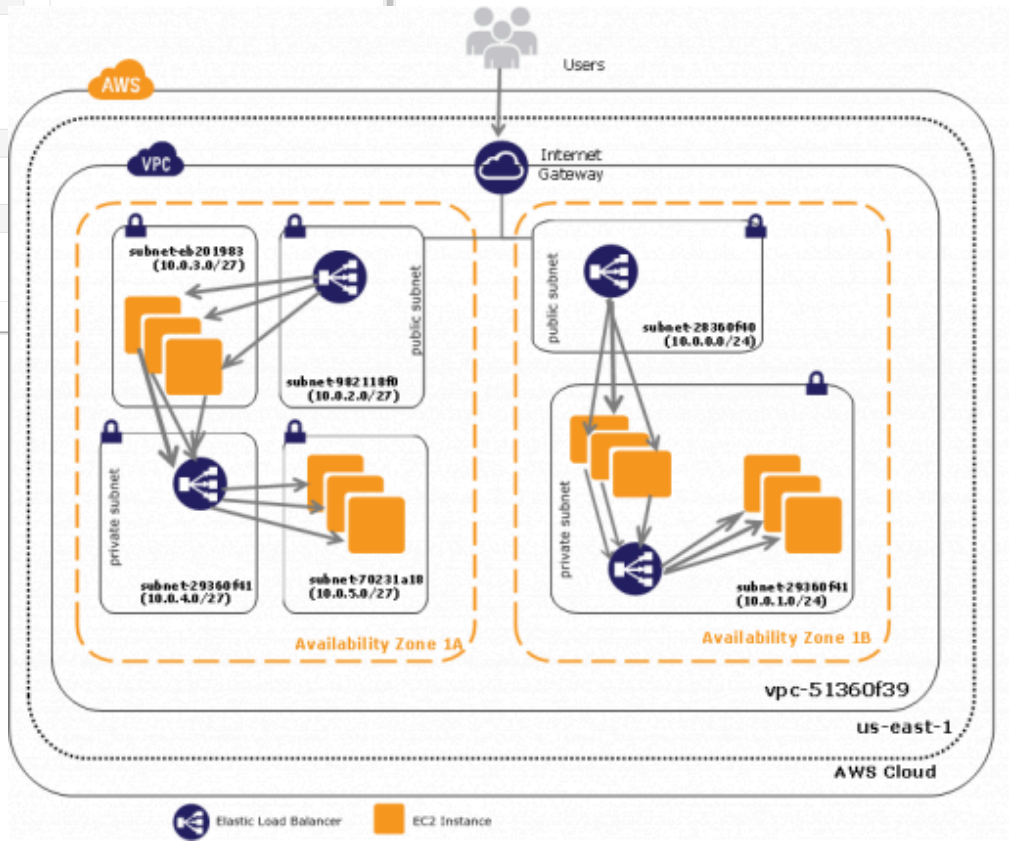
```
grunt install
```

Execute jetty via Grunt:

```
grunt jetty
```

Profiles available:

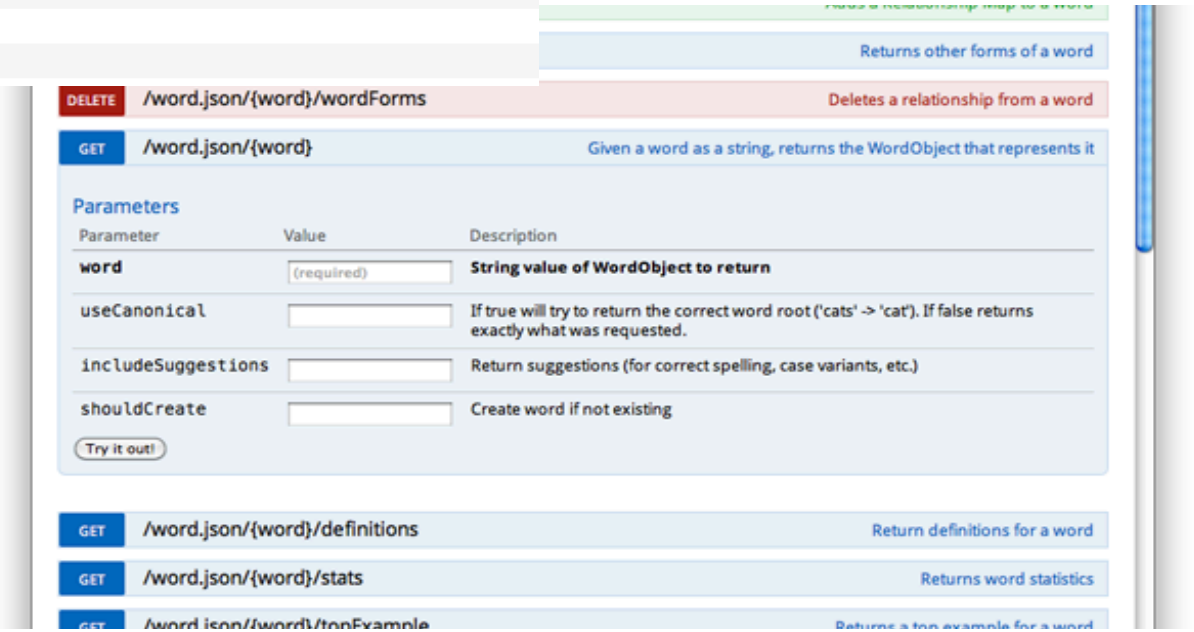
(Search for '@Profile(Profiles.PRODUCTION)' or '@Profile(Profiles.MOCK)')



API Docs with Swagger

```
@RequestMapping(value = "/news/{id}", method = RequestMethod.GET)
@ApiOperation(value = "Get a news item", notes = "Returns a news item")
NewsEntry entry(@PathVariable Long id) {
  return this.entries.get(id);
}

@RequestMapping(value = "/news/{id}", method = RequestMethod.POST)
@ApiOperation(value = "Update News", notes = "Updates a news item")
NewsEntry update(@RequestBody NewsEntry news) {
  this.entries.put(news.getId(), news);
  return news;
}
```



https://helloreverb.com/developers/swagger

Create a PACT

```
import java.util.HashMap;
import java.util.Map;

import static org.junit.Assert.assertEquals;

public class ExampleJavaConsumerPactTest extends ConsumerPactTest {

    @Override
    protected PactFragment createFragment(ConsumerPactBuilder.PactDslWithProvider builder) {
        Map<String, String> headers = new HashMap<String, String>();
        headers.put("testreqheader", "testreqheadervalue");

        return builder
            .given("test state") // NOTE: Using provider states are optional, you can leave it out
            .uponReceiving("java test interaction")
                .path("/")
                .method("GET")
                .headers(headers)
                .body("{\"test\":true}")
            .willRespondWith()
                .status(200)
                .headers(headers)
                .body("{\"responsetest\":true}").toFragment();
    }
```

https://github.com/DiUS/pact-jvm

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

Resource aware

Monitored thoroughly

Antifragile

High Cohesion / Loose Coupling (all the way down…)

- Code
- Architecture
  - Components
  - Services
- Public API
  - PayPal (bit.ly/1hnZNly)

Smashing the Monolith…

- Business functionality – “Cart Service”
- Technology chunk - “Email Service”
- Vertical Slice
  - “Registration” (Groupon: vimeo.com/105880150)
- Horizontal Slice
  - “User Repo” (Microservices: oreil.ly/1pp6qmx)

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

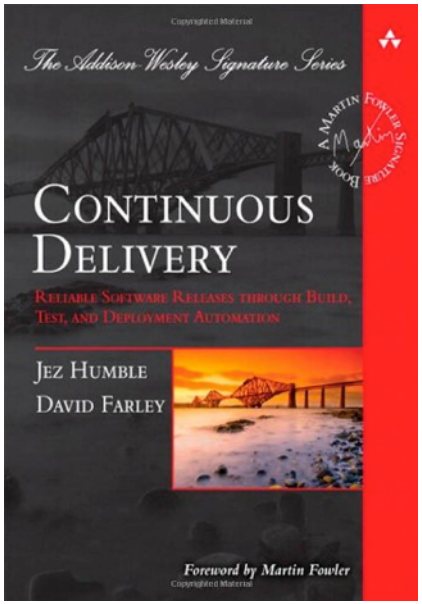
Resource aware

Monitored thoroughly

Antifragile

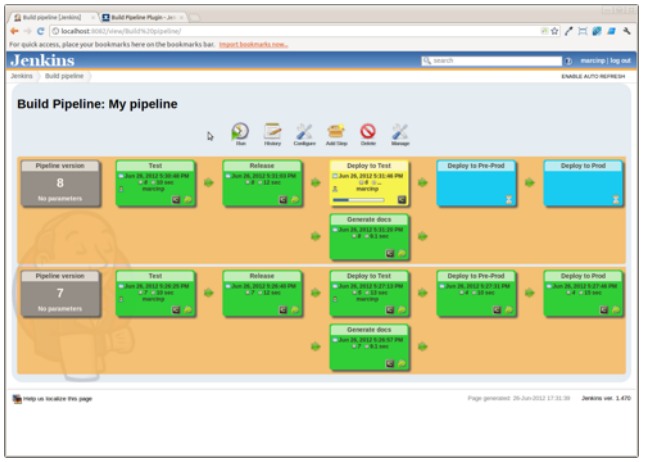
Automated from Commit to Cloud

- Continuous Integration
- Continuous Deployment
- Continuous Delivery



Our Build Pipeline

Jenkins, with plugins…



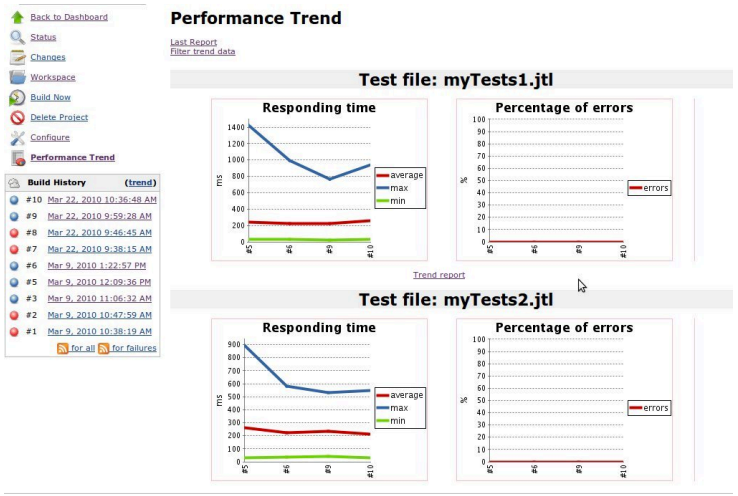
- Build Pipeline
  - wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin
- Parameterized build
  - wiki.jenkins-ci.org/display/JENKINS/Parameterized+Build
- Promoted Builds Plugin
  - wiki.jenkins-ci.org/display/JENKINS/Promoted+Builds+Plugin

Our Build Pipeline

- Component Build
  - Compile
  - Unit Tests (surefire)
  - Integration Tests (failsafe)
- Deployment onto QA Cloud
  - Python Scripts + Chef to provision
  - Verify success using Python

Our Build Pipeline

- Acceptance Tests
  - Cucumber (and Selenium)



- Performance Tests
  - Jmeter + Jenkins performance plugin
  - Make sure environment is realistic!!

- Live Deployment?

Automating QA

- Intra-component integration testing
  - Utilise embedded datastore/middleware
  - "Scassandra" (github.com/scassandra)
  - Service virtualisation (www.mbtest.org)
- Inter-component integration testing
  - The hardest part of SOA…
  - Consider 'synthetic txns'

Infrastructure: Say No To Snowflakes!

- Automate all provisioning (store in SCM)
- Fry...
  - Chef, Puppet, SaltStack, Ansible
  - Bash, Python (Fabric)
  - Vendor APIs
- …or bake?
  - Packer.io
  - Netflix Aminator

Infrastructure: Say No To Snowflakes!

- Doing "Proper Development"
  - Gareth Rushgrove at Craft Conf (bit.ly/1njuc49)
  - Chef Conf (www.youtube.com/user/getchef)

- Local tooling/testing
  - Vagrant (www.vagrantup.com)
  - Docker (www.docker.io)

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

Resource aware

Monitored thoroughly

Antifragile

Deployment Platform: What you've got…

What you think you want…

What you get…

Fact: 9 out of 10 cheetahs prefer the taste of an Ops team over tinned food

Thou Shalt Know thy Cloud…

- AWS “Magnetic” EBS 100 IOPS (by default)
  - My Mac SSD does 49K IOPS
- 1000Mbps network max transfer ~125MB/s
  - My Mac does 400+ MB/s Sequential Write to SSD
- “Noisy [virtual] Neighbours”

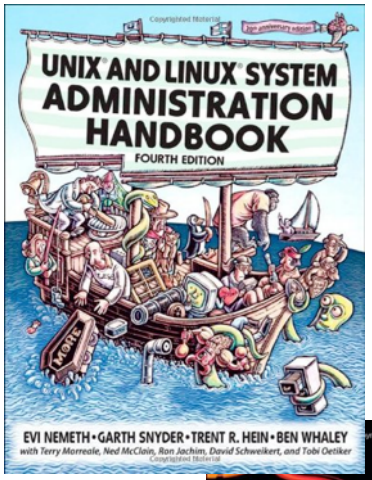
Reference for Mac statistics: bit.ly/1ftJZH8

Thinking/Acting Operationally

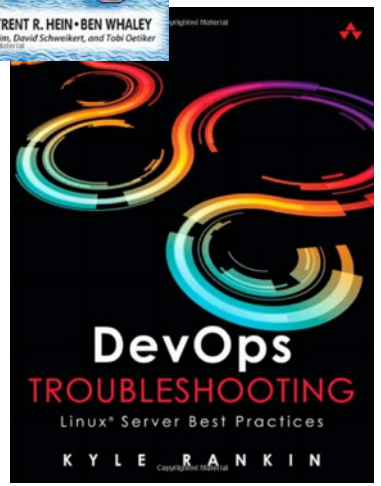
- Cultivate “Mechanical Sympathy”
- Virtualisation
  - Tech Target (bit.ly/1kDVqyG)
- Networking
  - ‘Unix and Linux System Administration Handbook’
  - aws.amazon.com/documentation

Thinking/Acting Operationally

- Learn Linux fundamentals



- Diagnostic skills
  - top, netstat, vmstat, tcpdump
  - Java utils: jps, jstat, jmap, jhat
  - "DevOps Troubleshooting" by K. Rankin



- Maybe grow a beard…

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

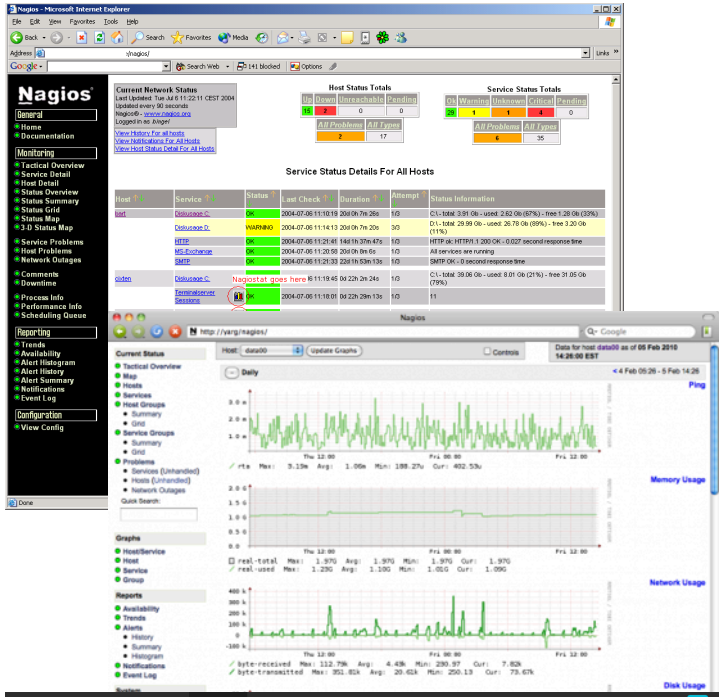
Resource aware

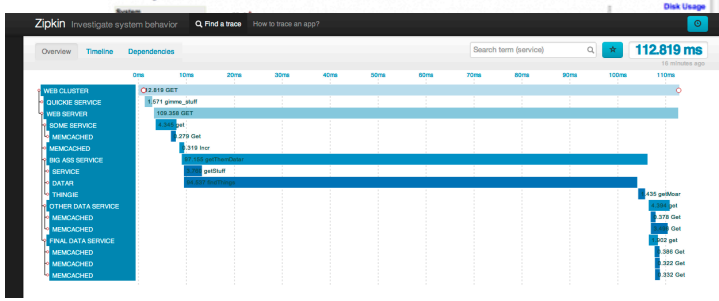
Monitored thoroughly

Antifragile

Monitor All The Things!

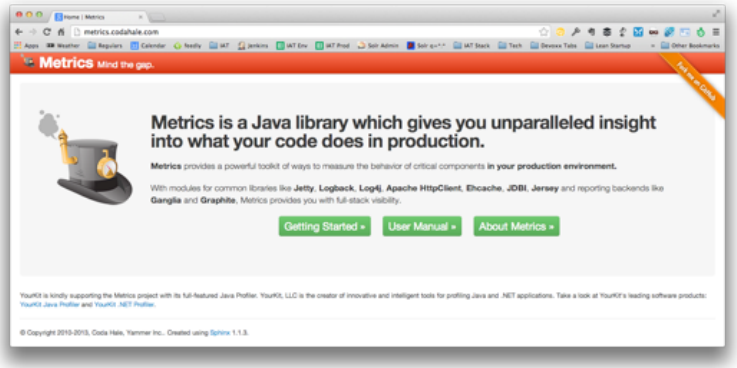
- Infrastructure monitoring
  - Nagios / Zabbix
  - AppDynamics
- Centralised Logging
- Distributed Tracing
  - twitter.github.io/zipkin





Component Metrics

- Dropwizard's Metrics
  - metrics.codahale.com



- Netflix's Servo
  - github.com/Netflix/servo

- Etsy's StatsD
  - github.com/etsy/statsd/wiki

Health Checks

Health Checks

Metrics also has the ability to centralize your service's health checks with the `metrics-healthchecks` module.

First, create a new `HealthCheckRegistry` instance:

```
final HealthCheckRegistry healthChecks = new HealthCheckRegistry();
```

Second, implement a `HealthCheck` subclass:

```
public class DatabaseHealthCheck extends HealthCheck {
    private final Database database;

    public DatabaseHealthCheck(Database database) {
        this.database = database;
    }

    @Override
    public HealthCheck.Result check() throws Exception {
        if (database.isConnected()) {
            return HealthCheck.Result.healthy();
        } else {
            return HealthCheck.Result.unhealthy("Cannot connect to " + database.getUrl());
        }
    }
}
```

Then register an instance of it with Metrics:

Gauges, Counters, Meters, Timers…

`static` field is fine.

Gauges

A gauge is an instantaneous measurement of a value. For example, we may want to measure the number of pending jobs in a queue:

```
public class QueueManager {
    private final Queue queue;

    public QueueManager(MetricRegistry metrics, String name) {
        this.queue = new Queue();
        metrics.register(MetricRegistry.name(QueueManager.class, name, "size"),
                         new Gauge<Integer>() {
                             @Override
                             public Integer getValue() {
                                 return queue.size();
                             }
                         });
    }
}
```

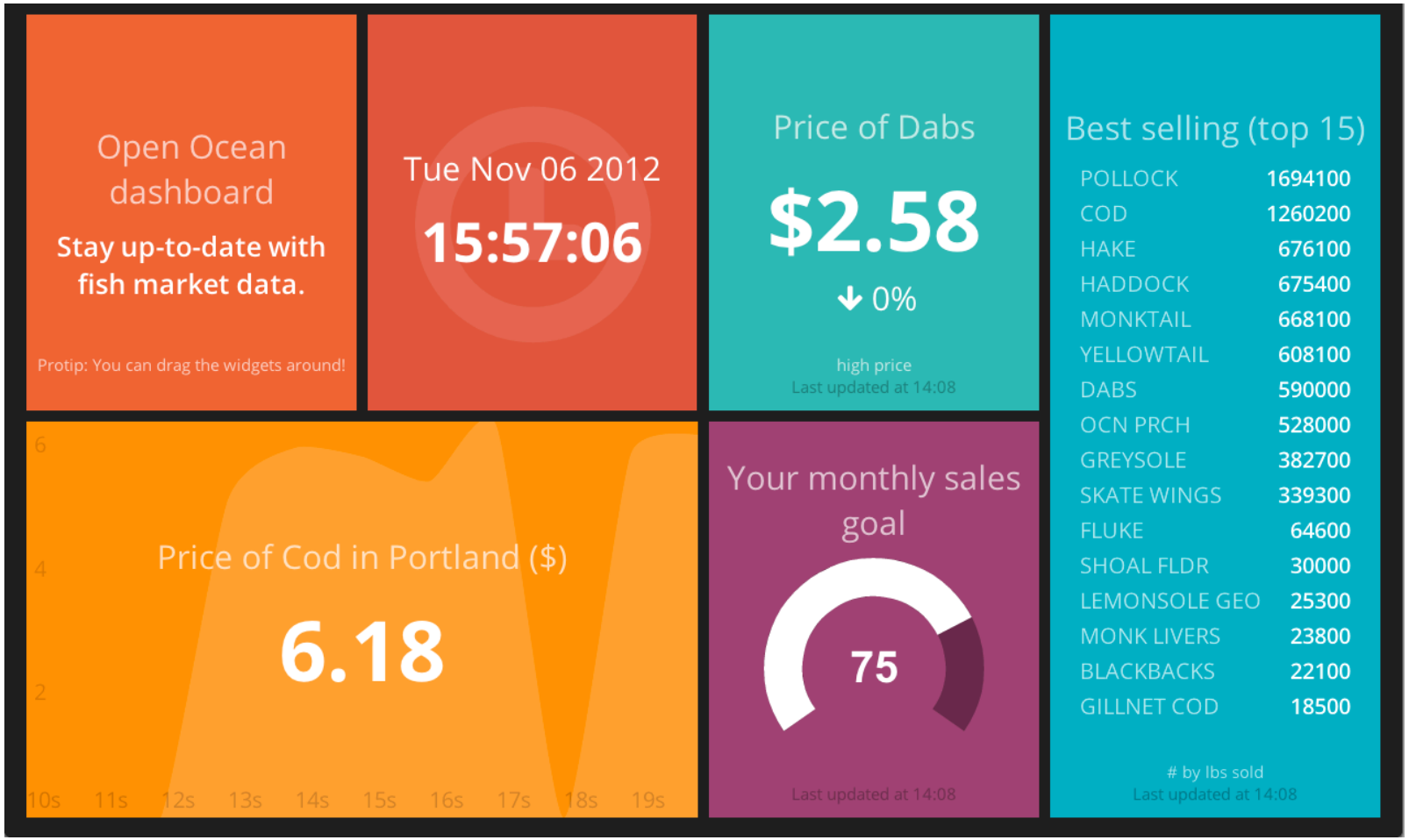
When this gauge is measured, it will return the number of jobs in the queue.

Every metric in a registry has a unique name, which is just a dotted-name string like `"things.count"` or `"com.example.Thing.latency"`. `MetricRegistry` has a static helper method for constructing these names:

```
MetricRegistry.name(QueueManager.class, "jobs", "size")
```

This will return a string with something like `"com.example.QueueManager.jobs.size"`.

Graph It!


Open Ocean dashboard
Stay up-to-date with fish market data.
Protip: You can drag the widgets around!
Tue Nov 06 2012
15:57:06
Price of Dabs
$2.58
0%
high price
Last updated at 14:08
Best selling (top 15)
POLLOCK 1694100
COD 1260200
HAKE 676100
HADDOCK 675400
MONKTAIL 668100
YELLOWTAIL 608100
DABS 590000
OCN PRCH 528000
GREYSOLE 382700
SKATE WINGS 339300
FLUKE 64600
SHOAL FLDR 30000
LEMONSOLE GEO 25300
MONK LIVERS 23800
BLACKBACKS 22100
GILLNET COD 18500
by lbs sold
Last updated at 14:08
Price of Cod in Portland ($)
6.18
6
4
2
10s 11s 12s 13s 14s 15s 16s 17s 18s 19s
Your monthly sales goal
75
Last updated at 14:08


IF IT MOVES, GRAPH IT . . .
imgflip.com


IF IT DOESN'T MOVE, GRAPH IT ANYWAY
IT MIGHT JUST BE TAKING A BREAK
imgflip.com

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

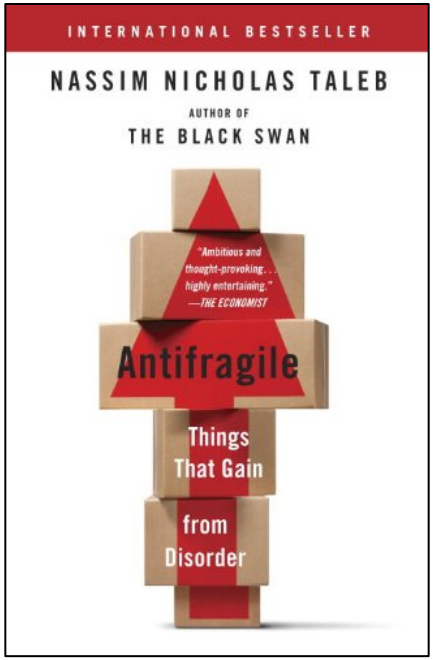
Resource aware

Monitored thoroughly

Antifragile

Antifragile

- The opposite of fragile?
  - Robust…
  - Antifragile…

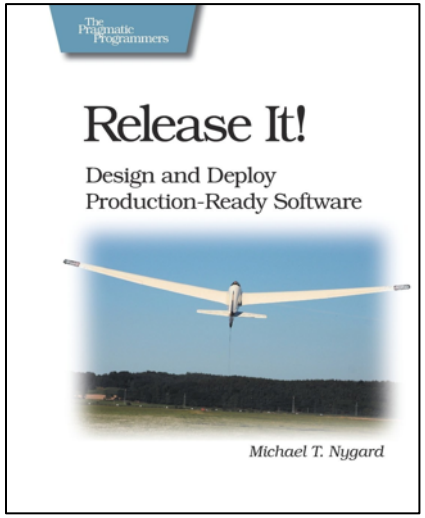


- Netflix are best-in-class
  - bit.ly/1gs5n3q

- System must be robust first!

Design for Failure

- Design patterns
  - Timeouts / retries
  - Bulkheads / circuit-breakers



- Inspiration
  - Chris Richardson (slidesha.re/1ft3vsg)
  - Netflix (bit.ly/1h5GMid)

Retries

```
Callable<Boolean> callable = new Callable<Boolean>() {
    public Boolean call() throws Exception {
        return true; // do something useful here
    }
};

Retryer<Boolean> retryer = RetryerBuilder.<Boolean>newBuilder()
        .retryIfResult(Predicates.<Boolean>isNull())
        .retryIfExceptionOfType(IOException.class)
        .retryIfRuntimeException()
        .withStopStrategy(StopStrategies.stopAfterAttempt(3))
        .build();
try {
    retryer.call(callable);
} catch (RetryException e) {
    e.printStackTrace();
} catch (ExecutionException e) {
    e.printStackTrace();
}
```

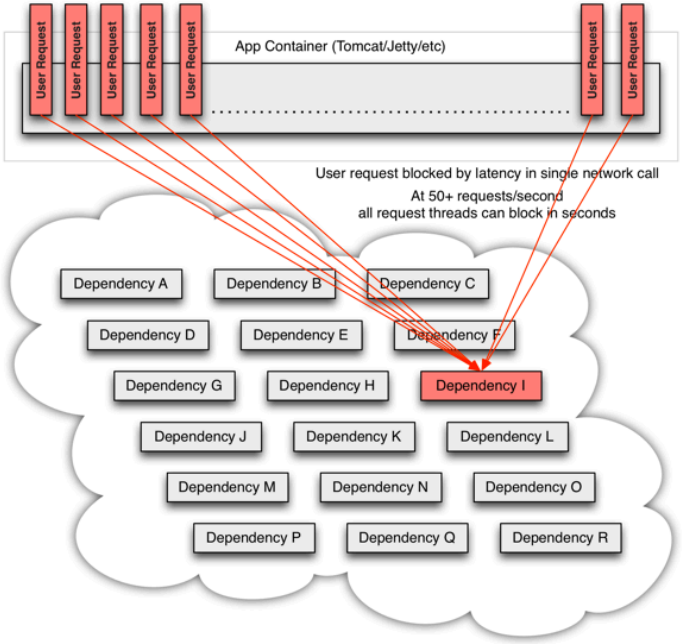
https://github.com/rholder/guava-retrying

Circuit-breaker

```
@HystrixCommand(fallbackMethod = "defaultUser")
public User getUserById(String id) {
    return userResource.getUserById(id);
}

@HystrixCommand(fallbackMethod = "defaultUserSecond")
private User defaultUser(String id) {
    return new User();
}

@HystrixCommand
private User defaultUserSecond(String id) {
    return new User("def", "def");
}
```



https://github.com/Netflix/Hystrix
https://github.com/Netflix/Hystrix/tree/master/hystrix-contrib/hystrix-javanica
http://java.dzone.com/articles/hystrix-and-spring-boot
http://projects.spring.io/spring-cloud/

Robust in the Cloud

- Distributed Computing Principles
  - 'For young bloods' (bit.ly/1pKVepz)

- Check your business goals
  - Lift and shift?
  - Cloud hybrid?
  - Cloud native? (Elastic scaling etc)

Antifragile Patterns: Respect the CAP

Eventual consistency (ACID vs BASE)

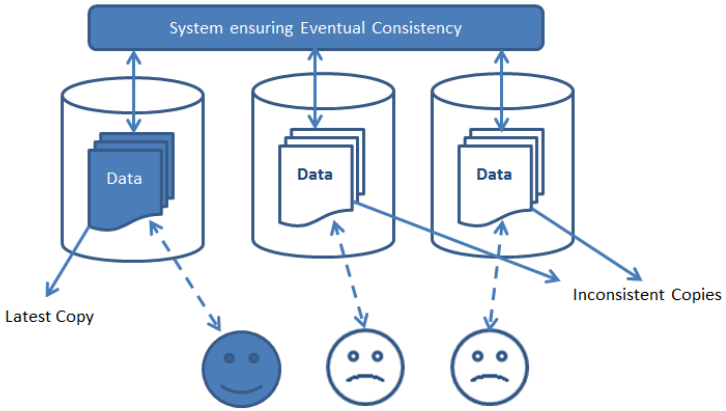


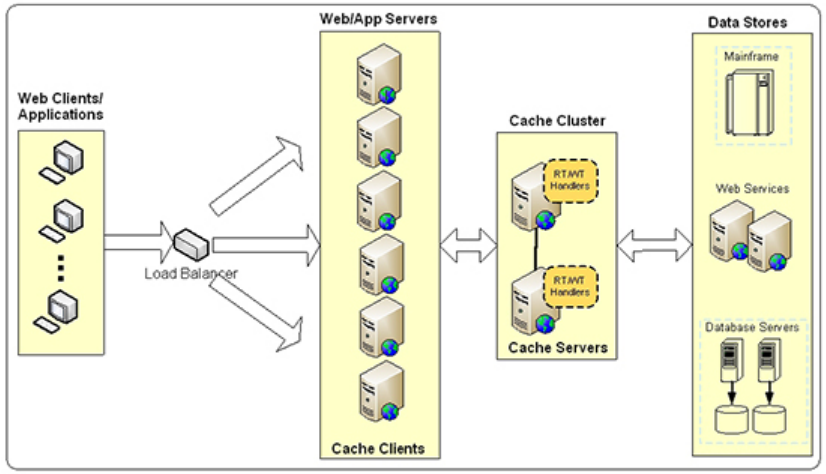
Figure showing a stale state where 2 copies of data are inconsistent with the latest one.

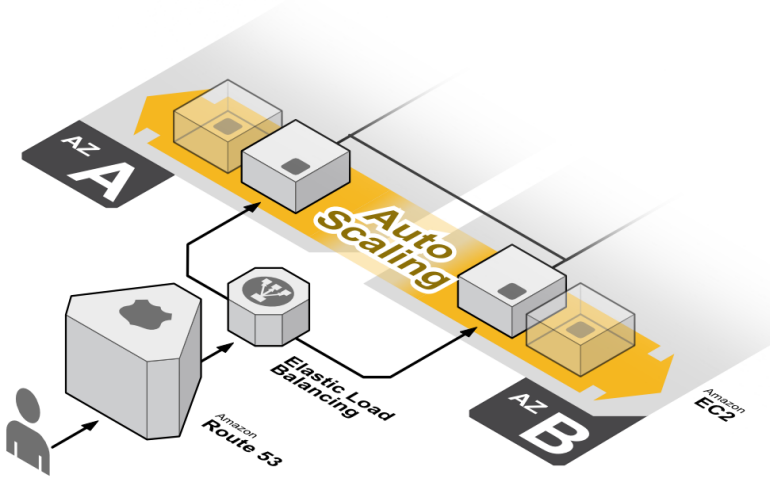
Clever caching (soft-state)

http://en.wikipedia.org/wiki/CAP_theorem
http://cloudshankar.blogspot.co.uk/2013/05/eventual-consistency.html

Antifragile Patterns: Elastic Scaling

Stateless components





Distributed data stores / caches

Asynchronous communication

Command Query Responsibility Segregation (CQRS)

So, Cloud Apps are ‘done’ when…

Documented (just enough)

Highly cohesive/loosely coupled (all the way down)

Automated from commit to Cloud

Resource aware

Monitored thoroughly

Antifragile

Thanks For Listening

- Massive thanks to all the Open Credo team!

- Questions / comments?
  - daniel.bryant@opencredo.com
  - @danielbryantuk