Building a Microservice Ecosystem: Some Assembly (Still) Required

Daniel Bryant

@danielbryantuk

OpenCredo

Agenda

- What is a microservice platform/ecosystem
- Local development of services can be hard
- Think about your build pipeline(s)
- Testing requires a paradigm shift
- Consider the infrastructure
- Plan for the inevitable…

Who Am I?

Principal Consultant at OpenCredo

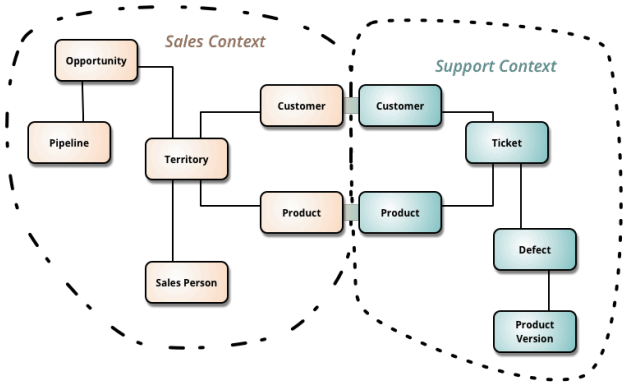
- Technical/digital transformation
- Java, Golang, CI/CD, DevOps
- Microservices, cloud and containers
- Maintainer of muservicesweekly.com

- London Java Community Associate
- Adopt OpenJDK and JSR
- InfoQ Editor, DZone MVB, Voxxed

So, What is a Microservice?

"Loosely coupled service oriented architecture with bounded contexts"

Adrian Cockcroft



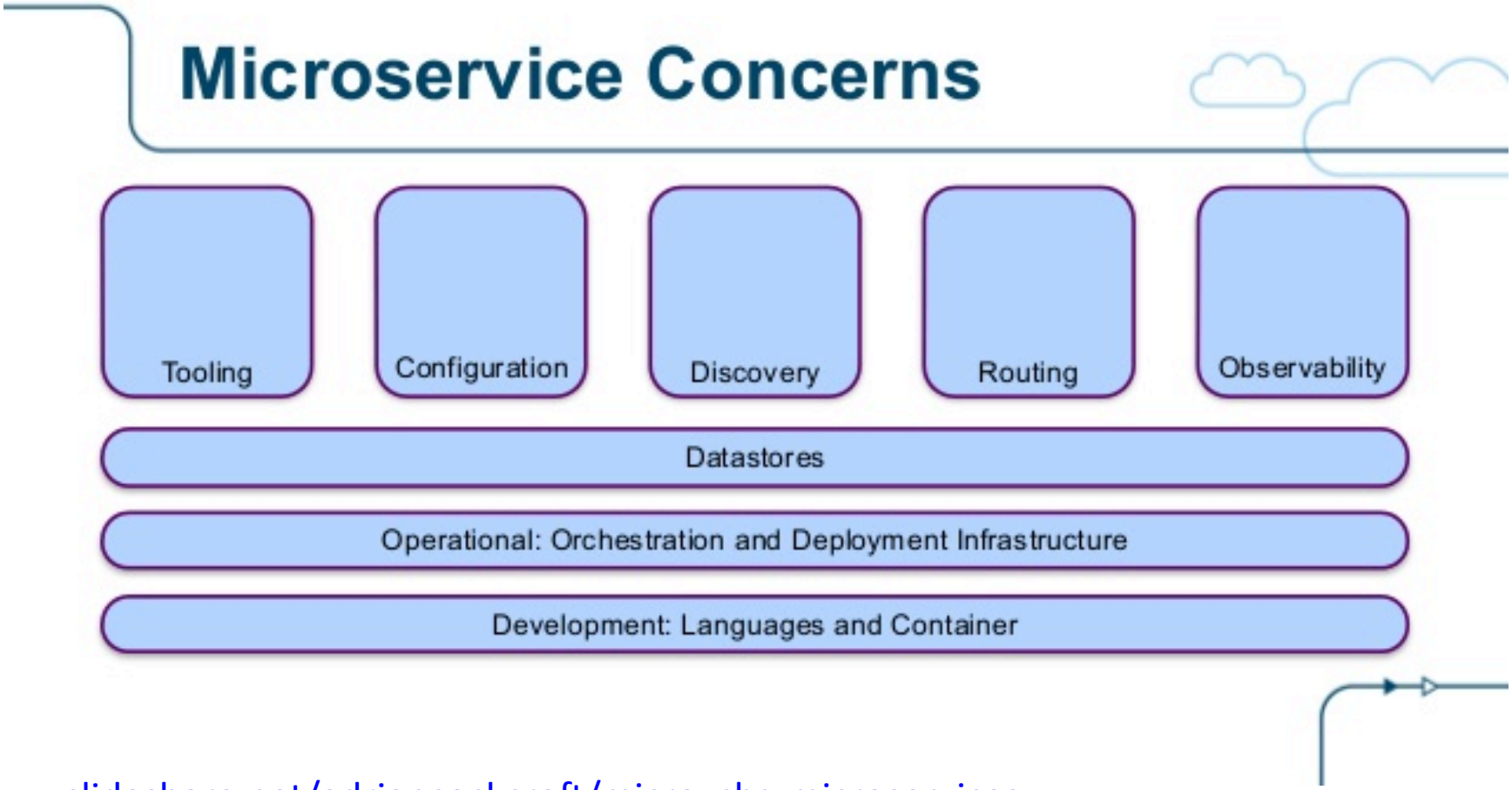
"Applications that fit in your head"

James Lewis

What do I mean by 'ecosystem'?

- Build
  - Local development, pipelines and integration
- Test
  - From local integration to E2E
- Deploy
- Operate
- Observe
  - Monitoring/logging/alerting

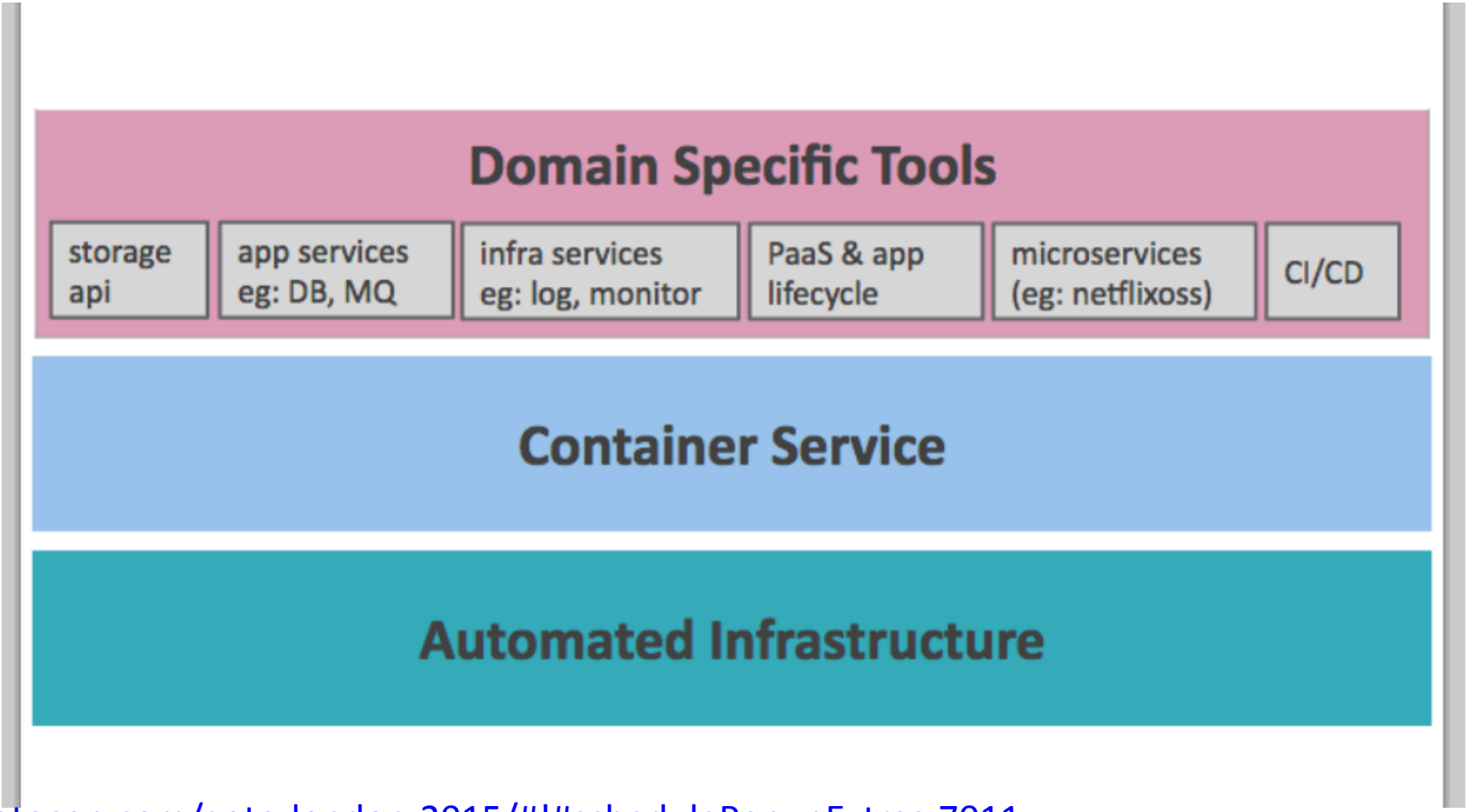
Adrian Cockcroft's Thoughts



www.slideshare.net/adriancockcroft/microxchg-microservices

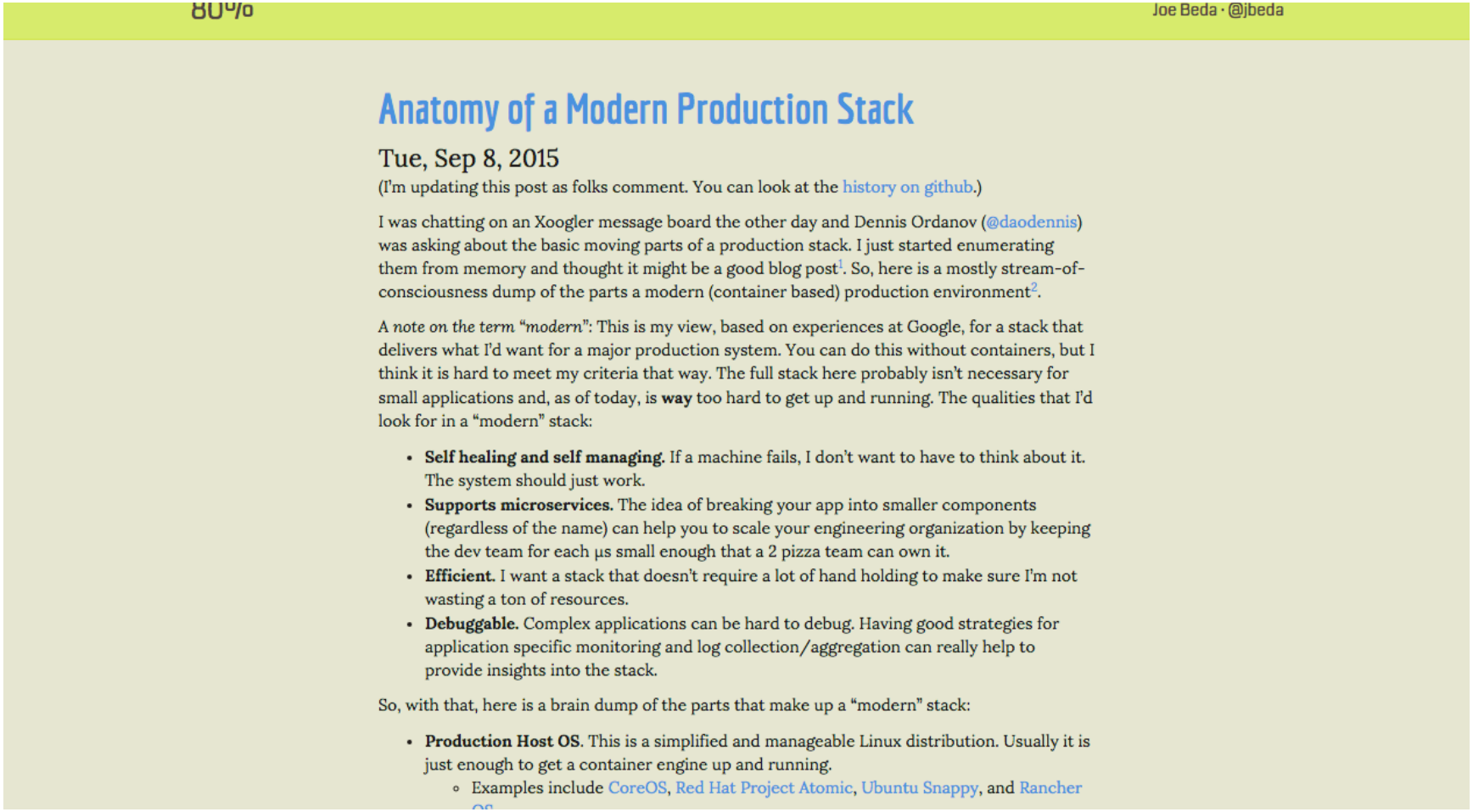
OpenCredo

Alexis Richardson's Thoughts



gotocon.com/goto-london-2015/#!#schedulePopupExtras-7011

Joe Beda's Thoughts



80%

Joe Beda · @jbeda

Anatomy of a Modern Production Stack

Tue, Sep 8, 2015

(I'm updating this post as folks comment. You can look at the history on github.)

I was chatting on an Xoogler message board the other day and Dennis Ordanov (@daodennis) was asking about the basic moving parts of a production stack. I just started enumerating them from memory and thought it might be a good blog post[1]. So, here is a mostly stream-of-consciousness dump of the parts a modern (container based) production environment[2].

A *note on the term "modern"*: This is my view, based on experiences at Google, for a stack that delivers what I'd want for a major production system. You can do this without containers, but I think it is hard to meet my criteria that way. The full stack here probably isn't necessary for small applications and, as of today, is **way** too hard to get up and running. The qualities that I'd look for in a "modern" stack:

- **Self healing and self managing.** If a machine fails, I don't want to have to think about it. The system should just work.
- **Supports microservices.** The idea of breaking your app into smaller components (regardless of the name) can help you to scale your engineering organization by keeping the dev team for each µs small enough that a 2 pizza team can own it.
- **Efficient.** I want a stack that doesn't require a lot of hand holding to make sure I'm not wasting a ton of resources.
- **Debuggable.** Complex applications can be hard to debug. Having good strategies for application specific monitoring and log collection/aggregation can really help to provide insights into the stack.

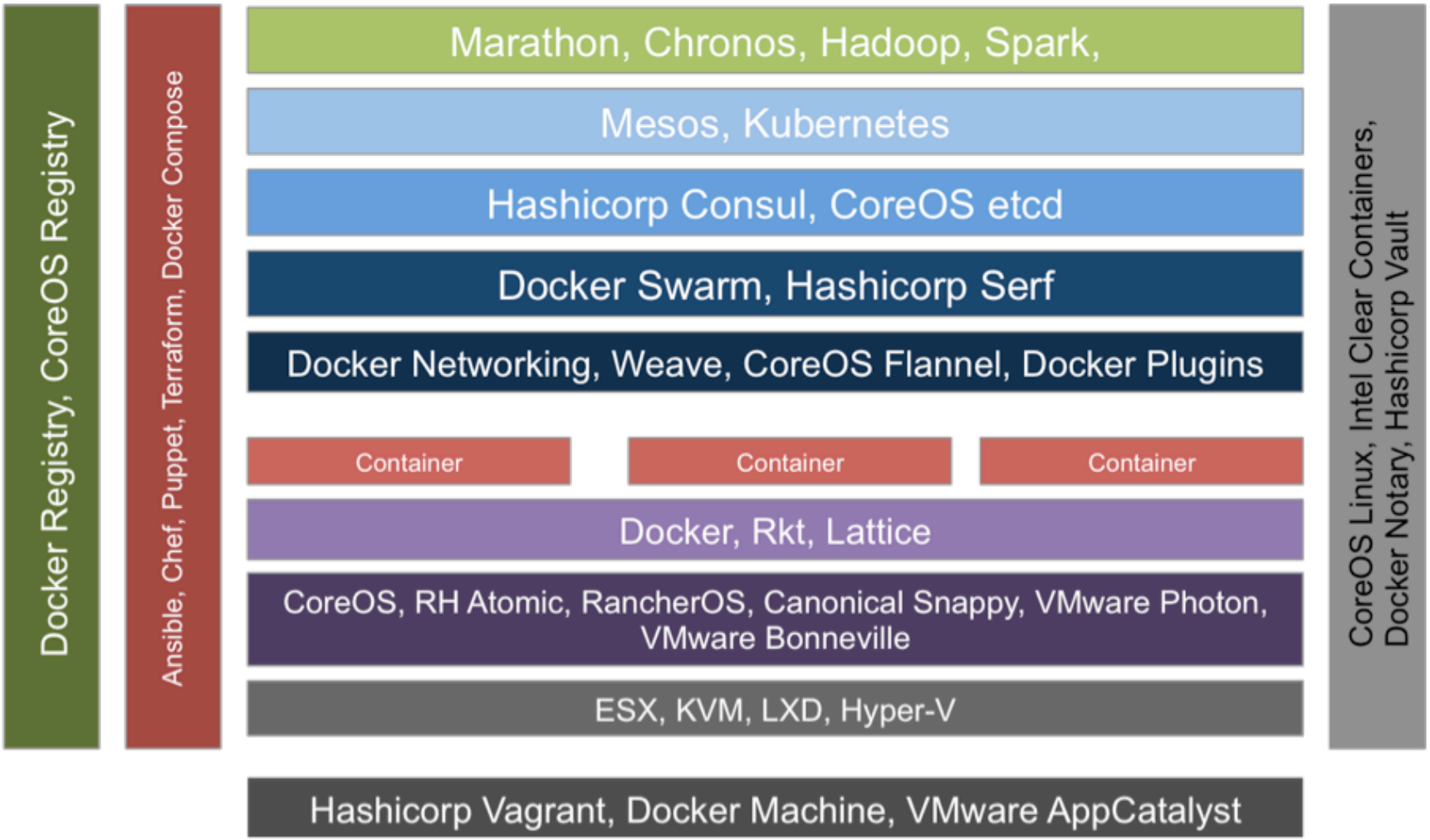
So, with that, here is a brain dump of the parts that make up a "modern" stack:

- **Production Host OS**. This is a simplified and manageable Linux distribution. Usually it is just enough to get a container engine up and running.
    - Examples include CoreOS, Red Hat Project Atomic, Ubuntu Snappy, and Rancher

www.eightypercent.net/post/layers-in-the-stack.html

OpenCredo

Technology Choices



wikibon.com/wp-content/uploads/container_implementations.png

OpenCredo

THE LOOK I GIVE

WHEN I'M COMPLETELY OVERWHELMED

Core (up front?) Decisions

- Platform
  - Cloud Foundry
  - Amazon ECS, Deis, Flynn
  - Docker Swarm (Tutum)
  - Kubernetes
  - Mesos
  - HashiCorp Nomad
  - DIY (maybe leverage Netflix's stuff?)

Core (up front?) Decisions

- Packaging
  - Services/tasks/jobs
  - ’buildpacked’
  - VMs
  - Containers

- Is Docker a de facto standard?…

Build

Developing Locally: The Basics

- GitHub's Boxen (Puppet)
- Pivotal's Sprout (Chef)
- Mac-dev-playbook (Ansible)

- Vagrant (Otto?)
- Docker Compose (Docker machine)

Developing Locally

- The naïve approach
  - Replication of env per service
  - …and dependencies and data stores and...
  - Soon gets crazy
- Local profiles + mocking/stubbing
  - Spring profiles + Mockito etc

Developing Locally

- Service virtualisation
  - Mountebank, Wiremock (Saboteur), Mirage
- 'Production-in-a-box' (IFTTT)
  - Docker Compose, Vagrant, cf_nise_installer
- Environment leasing
  - Create your own env (e.g. Hailo)

Developing Locally

OpenCredo
delivering emerging technology today

Consulting | Services | Clients | Blog | News | Company | Contact Us | EN | DE

Microservice Platforms: Some Assembly [Still] Required. Part Two

20 September, 2015
By Daniel Bryant

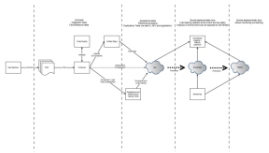
Tweet 67 | Share 31 | +1 1 | Like 7

Working Locally with Microservices

Over the past five years I have worked within several projects that used a 'microservice'-based architecture, and one constant issue I have encountered is the absence of standardised patterns for local development and 'off the shelf' development tooling that support this. When working with monoliths we have become quite adept at streamlining the development, build, test and deploy cycles. Development tooling to help with these processes is also readily available (and often integrated with our IDEs). For example, many platforms provide 'hot reloading' for viewing the effects of code changes in near-real time, automated execution of tests, regular local feedback from continuous integration servers, and tooling to enable the creation of a local environment that mimics the production stack.

The pre-pipeline (local) development process

If we look at a typical build pipeline for a monolithic application we can see that the flow of a single monolithic component is relatively easy to orchestrate. The pipeline obviously gets more complex when we are dealing with microservices, and we'll cover this in future blog posts, but for now we will look at the pre-pipeline local development phase that will most likely involve working simultaneously with multiple dependent services.



When working with a monolithic codebase we should be able to assume that the creation and configuration of a local development machine is as least as easy to configure as a QA environment (and if it isn't, then you should be asking why!). Local developer machine configuration tooling such as Github's Boxen (Puppet), Pivotal's Sprout (Chef), or mac-

Search OpenCredo

Google™ Custom Search | Search

Tweets | Follow

OpenCredo @OpenCredo 1h
Catch @danielbryantuk at JavaOne for talks on: #Microservices, debugging & building better software ow.ly/TzbEi #JavaOne @docker
Expand

OpenCredo @OpenCredo 1h
Microservices weekly #2 is out, curated by @danielbryantuk #microservices, #ModularMonoliths #Security @Docker ow.ly/TzaZ8
Expand

Daniel Bryant @danielbryantuk 3h
Microservices Weekly #2 newsletter: Modular monoliths, securing microservices, and unikernels -> buff.ly/1Lw3v6t via @OpenCredo
Retweeted by OpenCredo
Expand

ContainerSched 2015 @containersched 16 Oct
We're so excited that incredible John Wilkes will host keynote at inaugural Container &Scheduler Tech Conf. Are you? pic.twitter.com/U3rnSDFPZr
Retweeted by OpenCredo

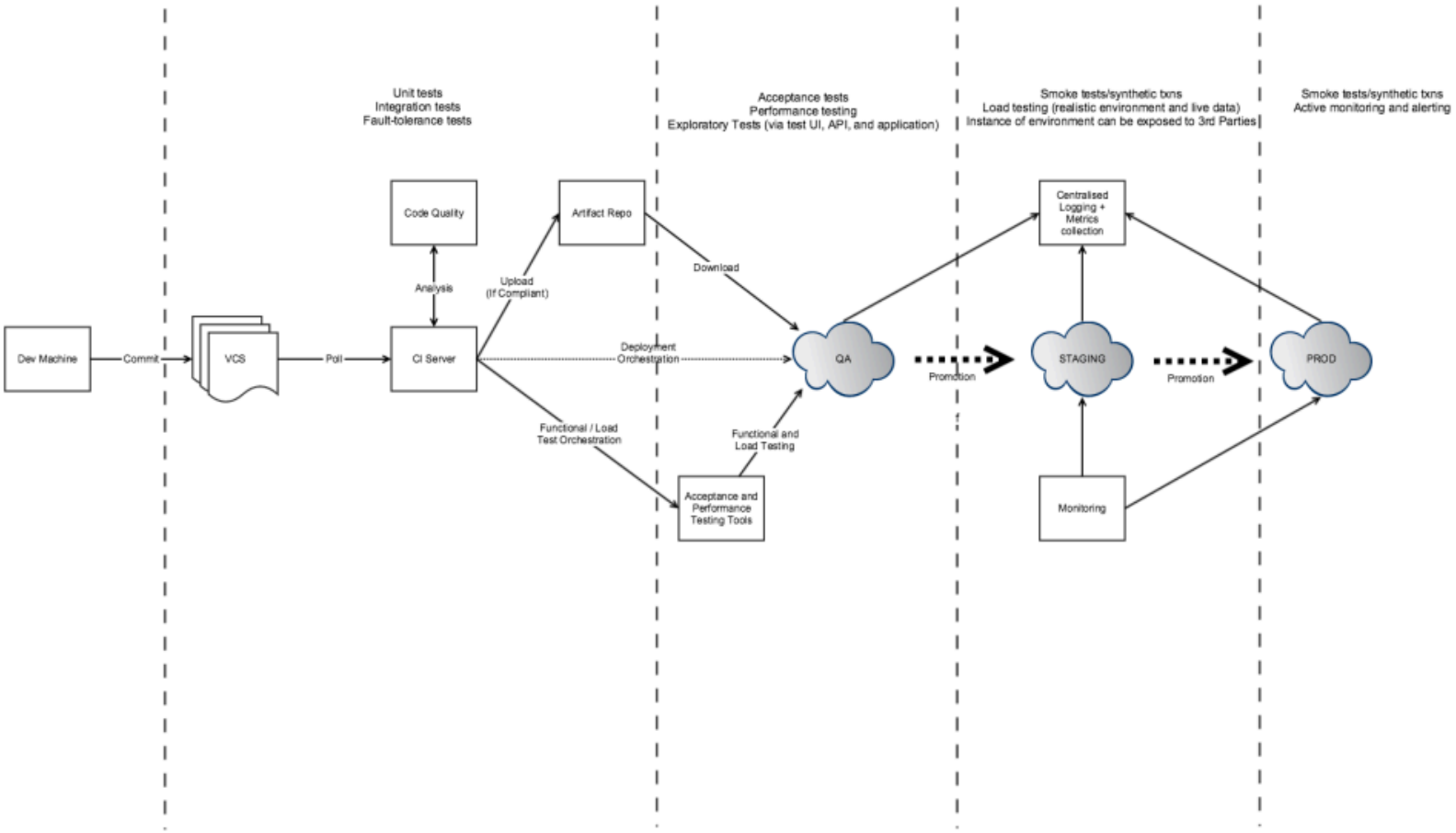
Tweet to @OpenCredo

Recent Blogs

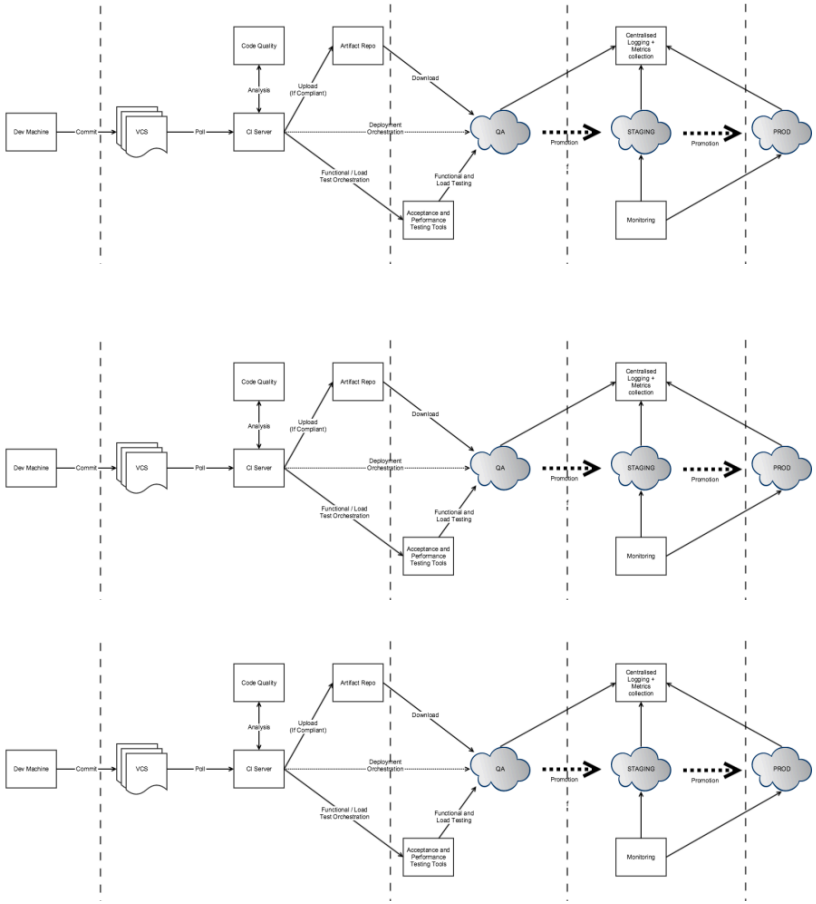
www.opencredo.com/2015/09/20/working-locally-with-microservices/

OpenCredo

Create a Pipeline



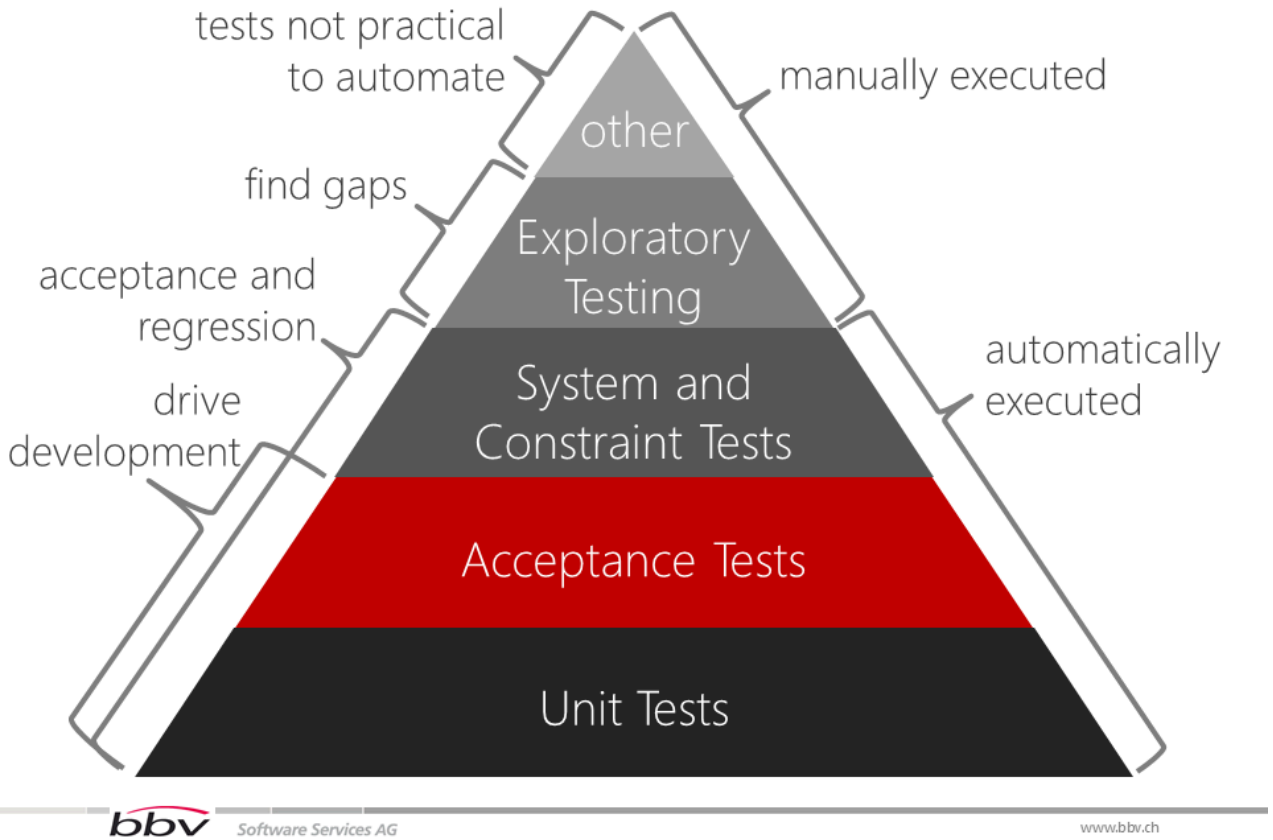
Multi-service / Multi-pipeline?



?

Test

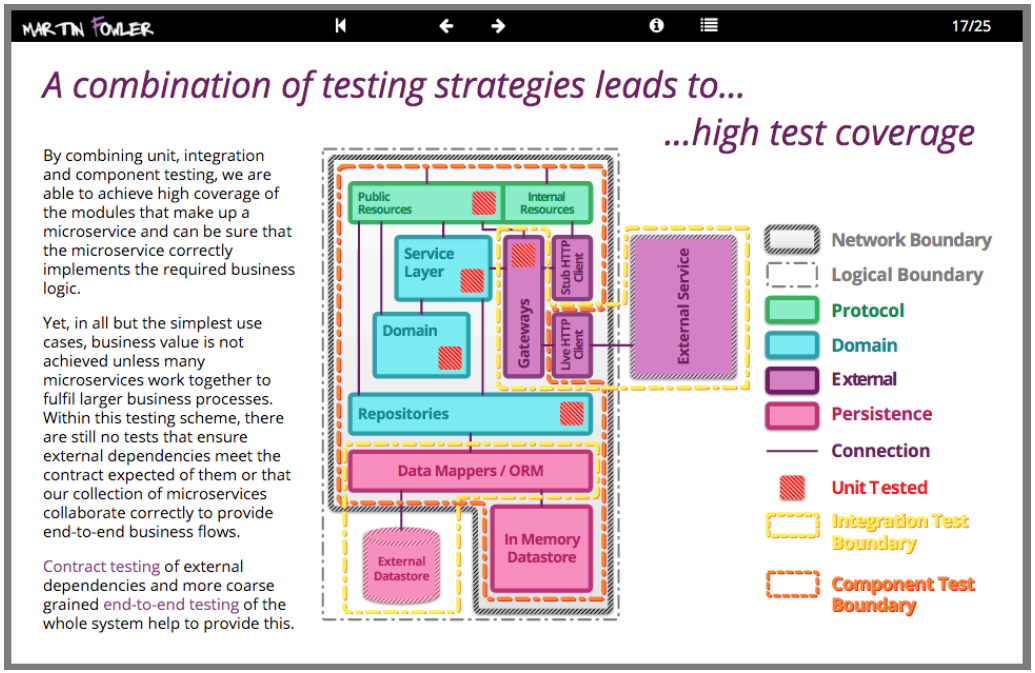
Always Remember…



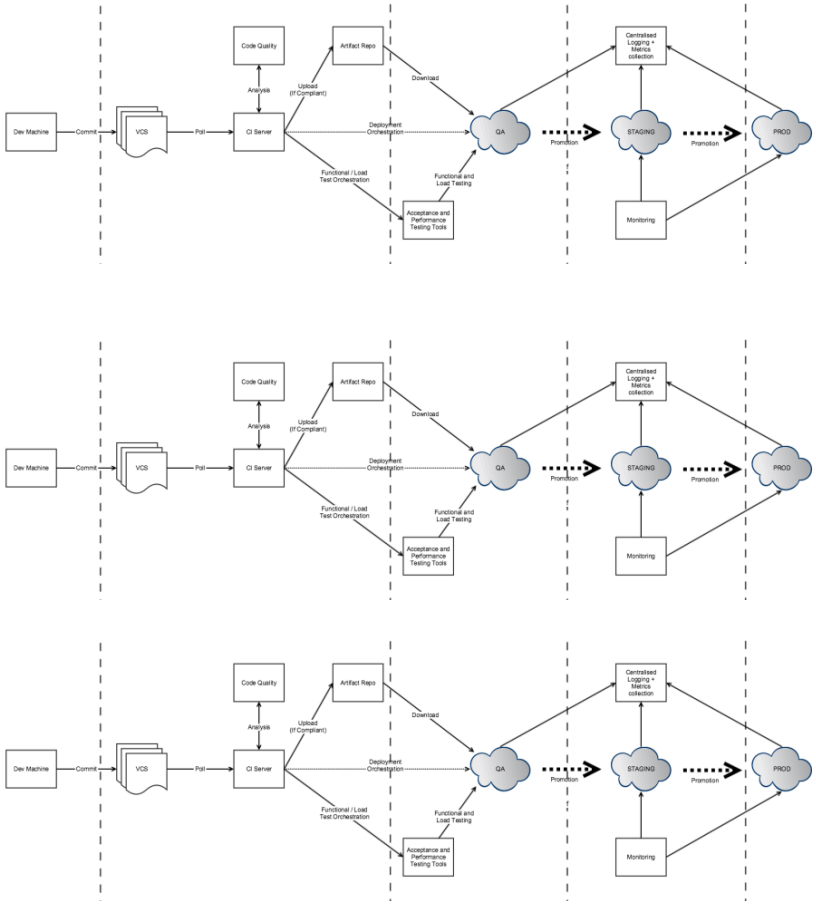
blog.bbv.ch/2012/06/13/acceptance-test-driven-development/

Testing Basics

- Toby Clemson's article
  - martinfowler.com/articles/microservice-testing



Multi-service / Multi-pipeline?



?

Pipeline

- Multiple pipelines?
  - Beware of rubber stamping (distributed monolith)
  - 'semver' if you must (semver.org) e.g. 1.2.1
- Shared staging
  - Critical path testing (why not prod?)
- Ensured consistency
  - Consumer-based contracts
  - Backwards compatibility

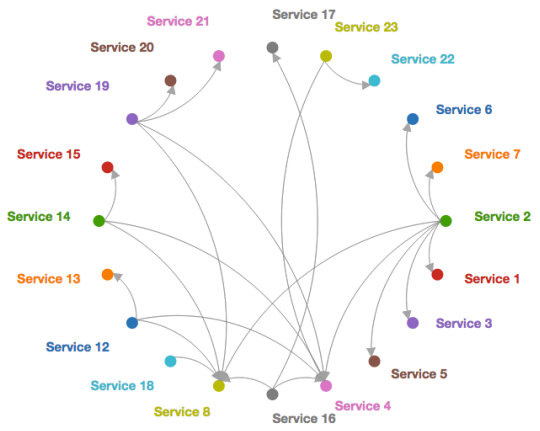
Integration Testing

- Contracts
  - Pact-JVM github.com/DiUS/pact-jvm
  - PACT broker github.com/bethesque/pact_broker
  - Examples: github.com/mstine/microservices-pact

HAL Browser

Relationships

Consumer		Provider
Service 10	→	Service 11
Service 10	→	Service 24
Service 10	→	Service 9
Service 12	→	Service 13
Service 12	→	Service 4
Service 12	→	Service 8
Service 14	→	Service 15
Service 14	→	Service 4
Service 14	→	Service 8



OpenCredo

My Opinions

- BDD services API
  - Serenity BDD
- Contract tests (failure is a conversation)
- Component test and unit test (as normal)
  - Maven surefire/failsafe
- BDD critical paths throughout application
  - Including API journey

Final Words on Testing

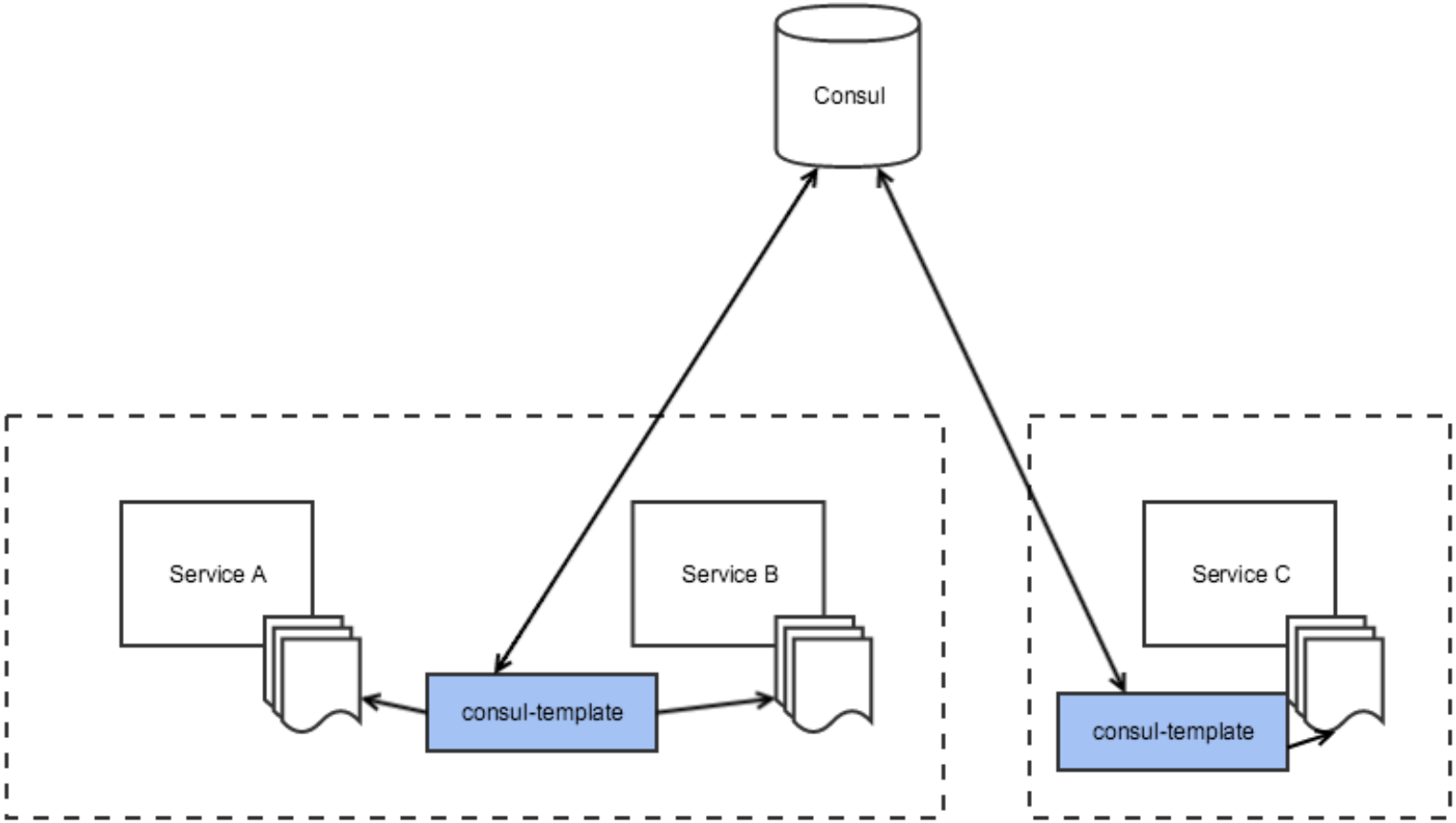
- Don't forget the "ilities"
- Security / reliability
  - ZAP (from the OWASP team)
  - github.com/continuumsecurity/bdd-security
- Performance / scalability
  - Jmeter (Jenkins Performance plugin)
  - flood.io

Deploy

Separate Deploy and Release

- Feature flags
  - Difficult at scale (and distribution)
  - Enable at ingress
- Incremental (phased) rollout
- Canary vs blue/green
- Avoid datastore migrations (if possible)

Centralise Configuration



Centralise Configuration

- Consul & consul-template
- Etcd/ZK & confd
- Netflix Archaius
- Spring Cloud

- Watch out for application rollback!

Operate

Building Blocks

- Standardise on an OS
  - Amazon Linux vs mainstream distros
- HashiCorp Terraform
  - VMWare? cloud-init and vcloud-tools
  - "Boot my secure government cloud"
- "CAPS"
  - Chef, Ansible, Puppet, SaltStack
  - Automated sysadmin

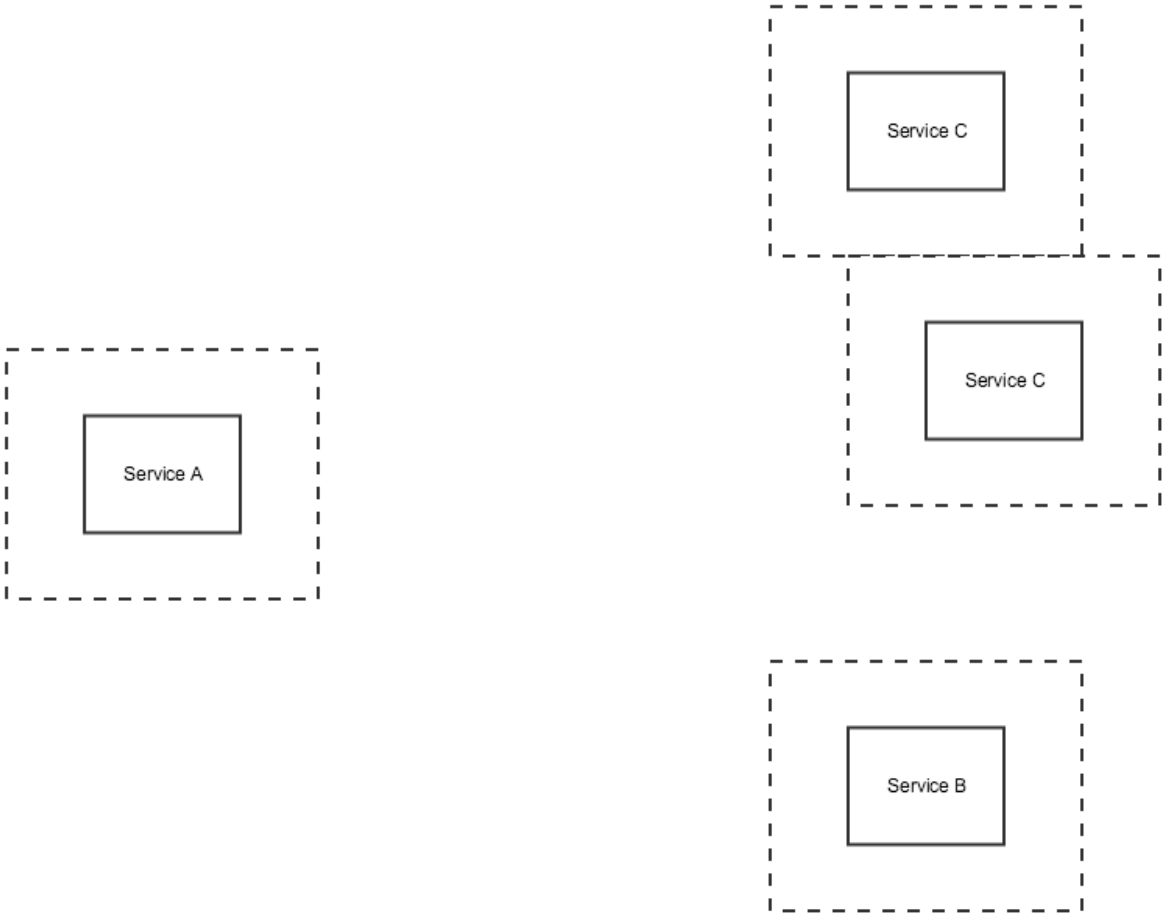
DevOps and Programmable Infra

- It involves programming…
- Introduce SOLID principles
- Good CI/CD principles
  - Gitflow etc

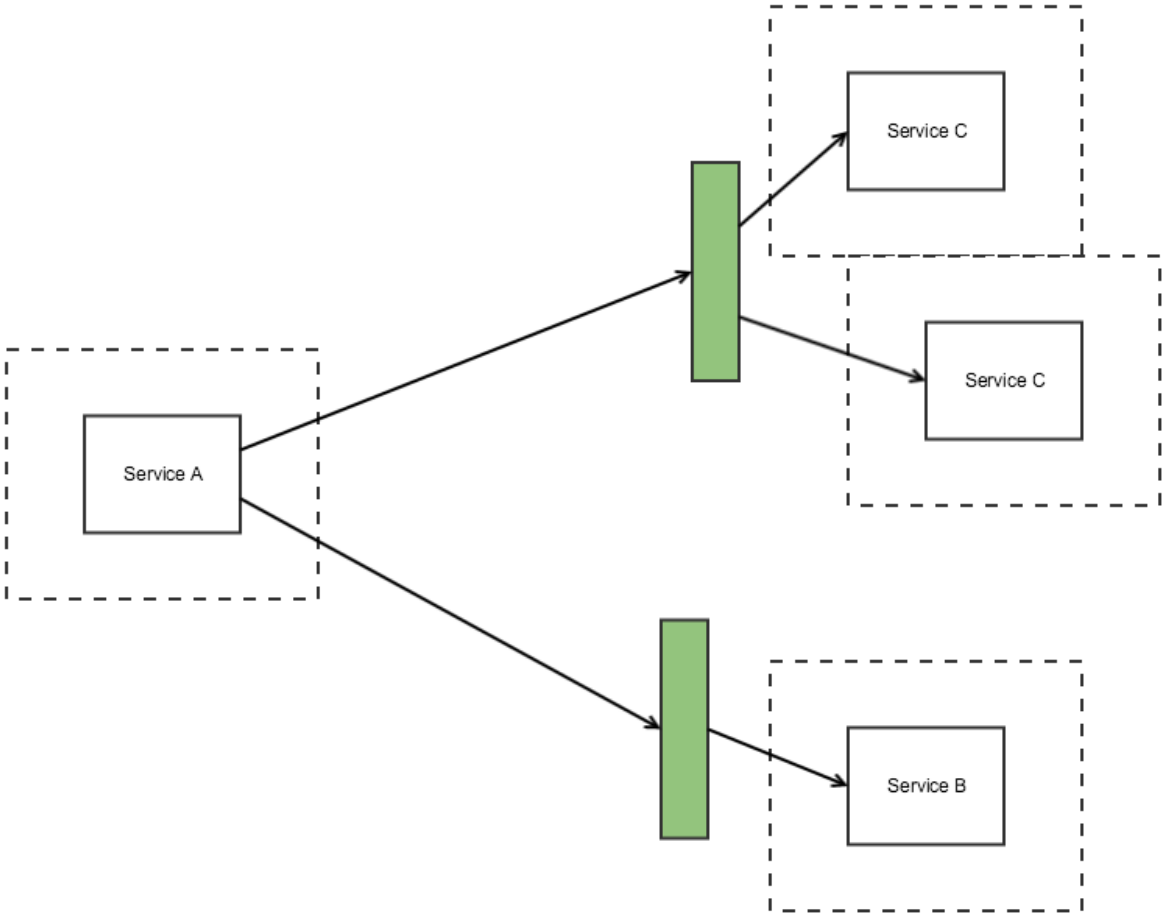
Service Discovery

- External
  - HAProxy / nginx / ELB etc
- Client-side
  - Netflix Ribbon (with Prana)
  - ‘Baker Street’ (extending SmartStack)
  - srv-router
- Kubernetes and CF are good to go

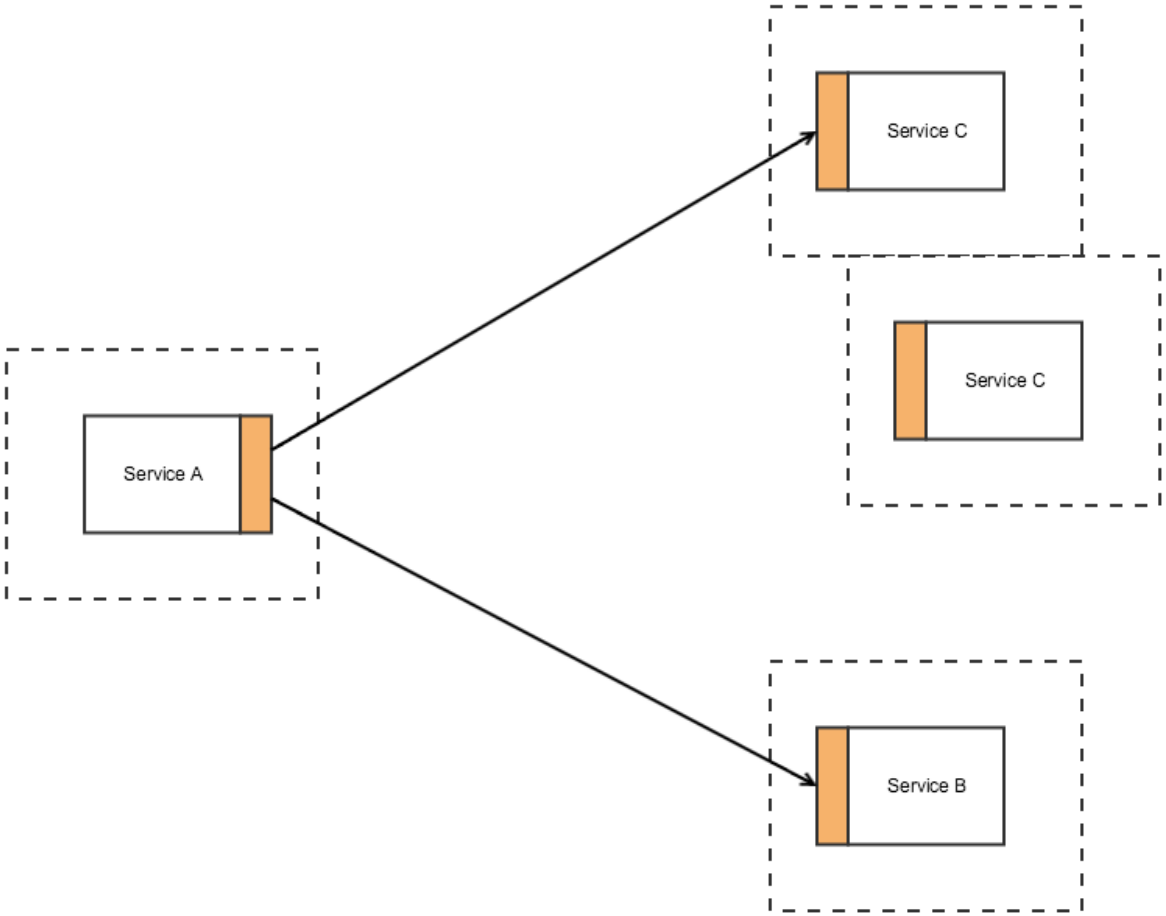
Service Discovery



External



Client-side

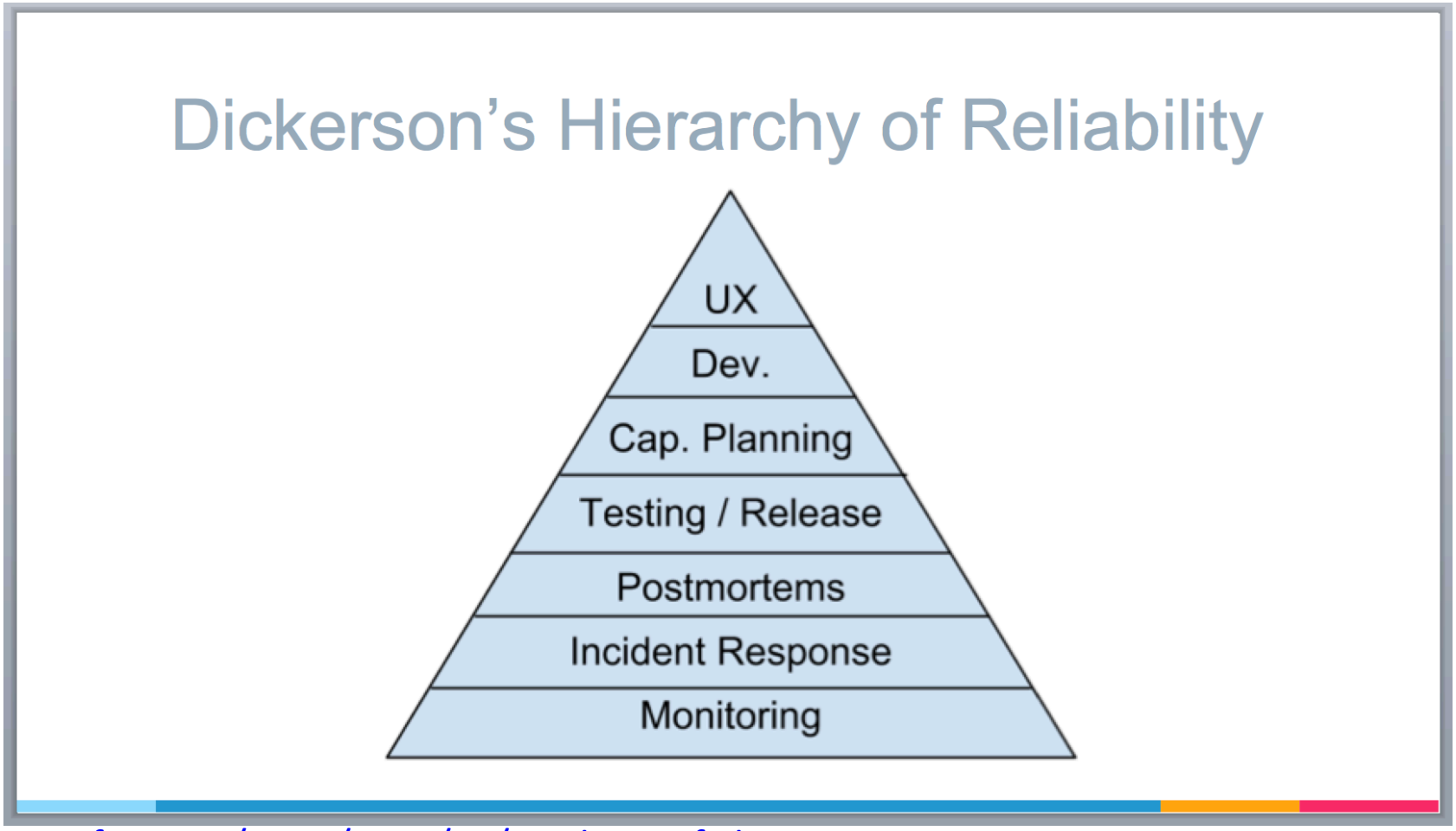


Service Discovery

- External
  - HAProxy / nginx / ELB etc
- Client-side
  - Netflix Ribbon (with Prana)
  - ‘Baker Street’ (extending SmartStack)
  - srv-router
- Kubernetes and CF are good to go

Observe

Monitoring and People



www.infoq.com/news/2015/06/too-big-to-fail

Start with the Basics

- Health checks
  - Coda Hale (DropWizard) Metrics
  - Spring Boot actuator
- KPIs for apps (and business)
  - Assertions / invariants
  - Throughput
  - Queue length

Logging

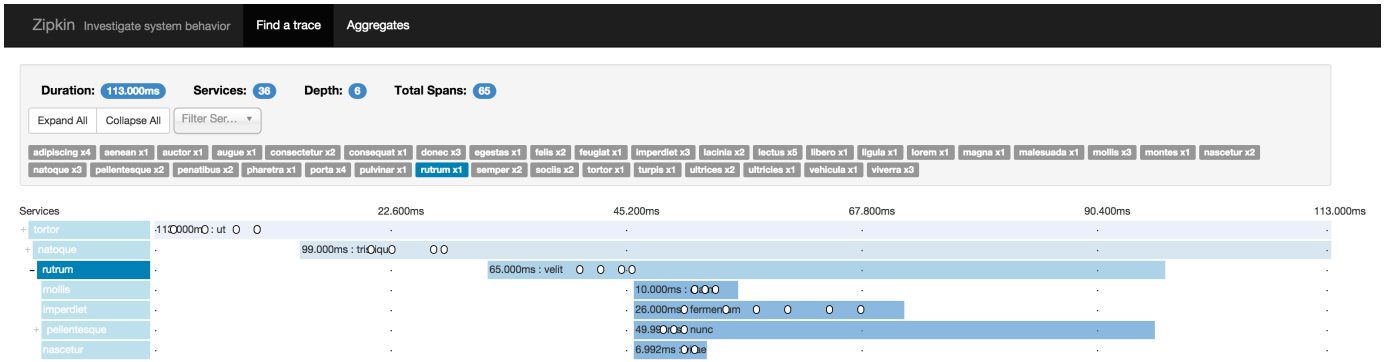
- "What every engineer should know"
- "10 Tips for Proper Application Logging"
- ElasticSearch-Logstash-Kibana (ELK)
  - Buffer/proxy log sending or…
  - Mount directory into container

Kibana FTW


APPTRACK Search
a day ago to a few seconds ago
apptrack_status=APPTRACK_SESSION_CREATE
QUERY
time must
field : @timestamp
from : now-24h
to : now
FILTERING
HISTORY
HISTOGRAM
View | Zoom Out | Create (1513068) count per 10m | (1513068 hits)
PROTOCOLS
TERMS
junos-http (637076) junos-https (389881) junos-dns-udp (255511) None (124013)
junos-winframe (44492) junos-ldap (14974) icmp (12231) junos-dns-tcp (10711) junos-smb (6136)
junos-smtp (5155) Other values (12906)
junos-http 42%
junos-dns-udp 17%
junos-https 26%
SOURCE IP
10.200.22.75 (148769) 10.22.2.38 (146608) 10.22.2.37 (75788) 10.32.5.99 (67955)
10.200.22.18 (64128) 10.22.2.16 (23671) 10.22.112.21 (21849) 10.22.3.4 (17152)
10.22.124.11 (16904) 10.22.113.30 (15306)
DEST IP
10.22.2.37 (64473) 10.22.2.41 (56064)
10.22.2.52 (27737) 10.22.2.51 (27492)
10.22.2.55 (19955) 10.22.2.54 (18684)
ALL EVENTS
TABLE
0 to 100 of 500 available for paging
@timestamp source_ip source_port dest_ip dest_port protocol apptrack_status
2014-01-30T14:38:50.000+11:00 10.32.5.99 54330 443 junos-https APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.22.113.42 64735 80 junos-http APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.22.16.250 50093 443 junos-https APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.22.2.38 60371 53 junos-dns-udp APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.22.2.38 60949 53 junos-dns-udp APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.22.2.38 61303 53 junos-dns-udp APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.200.22.75 63061 10.22.2.54 2598 None APPTRACK_SESSION_CREATE
2014-01-30T14:38:50.000+11:00 10.200.22.75 47272 10.22.2.51 2598 None APPTRACK_SESSION_CREATE

Distributed Tracing

- github.com/daniel-bryant-uk/correlation-id-async
- MDC logging (logback.qos.ch/manual/mdc.html)
- Zipkin/Brave (github.com/openzipkin/docker-zipkin)



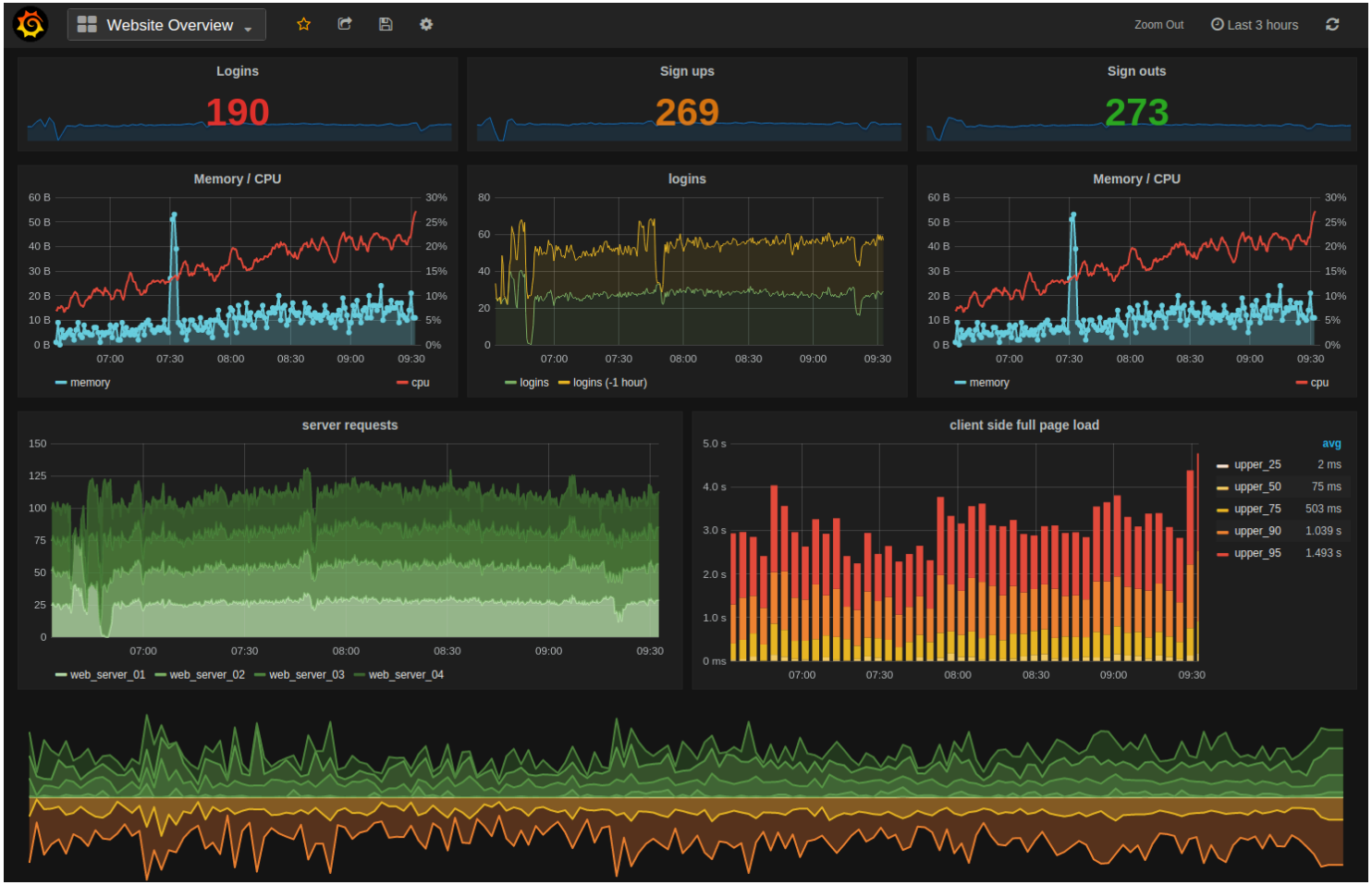
OpenCredo

Monitoring

- Push
  - Spring Boot actuator e.g. InfluxDbExporter
- Pull
  - E.g. Telegraf (shout to Tareq Abedrabbo)
- InfluxDB vs prometheus vs graphite vs opentsdb
- Information radiators
  - Aggregate vs individual

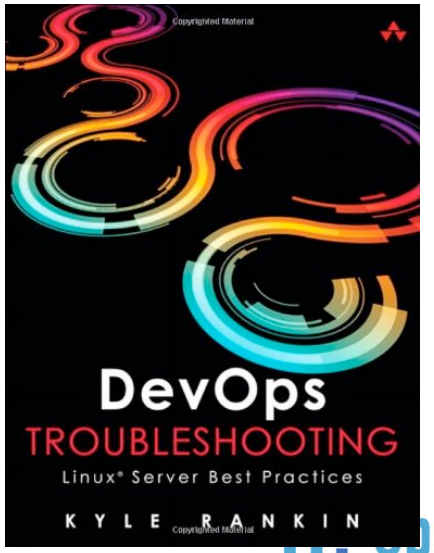
Aggregation: Sick Cattle, Not Sick Pets

Grafana FTW


Website Overview
Zoom Out
Last 3 hours
Logins
190
Sign ups
269
Sign outs
273
Memory / CPU
logins
Memory / CPU
memory
cpu
logins
logins (-1 hour)
server requests
web_server_01
web_server_02
web_server_03
web_server_04
client side full page load
avg
upper_25 2 ms
upper_50 75 ms
upper_75 503 ms
upper_90 1.039 s
upper_95 1.493 s

Problems?

- Rob Ewaschuk's "Philosophy on Alerting"
- Brendan Gregg's USE method
  - "check utilization, saturation, and errors."
- "DevOps Troubleshooting"
  - Kyle Rankin



Summary

- Looked at “microservice platforms”
- Local development of 3+ services is hard
- Think about your build pipeline(s)
- Testing requires a paradigm shift
- Consider the infrastructure
- Plan for the inevitable issues…
- Try not to create your own platform...


Colin Humphreys
@hatofmonkeys
Following
Started writing my own scheduler, router, health management, deployer, container builder, logging infra, cli, ...
It looks like you're trying to build your own platform. Would you like to do something useful instead?
use Cloud Foundry so you can focus on creating actual value
waste millions building your own platform you're special in your own way
RETWEETS
47
FAVORITES
31
8:29 AM - 13 Oct 2015

Thanks

Feedback is always welcomed!

daniel.bryant@opencredo.com

@danielbryantuk

www.opencredo.com

www.muservicesweekly.com