Microservizi: runtime, metodi, pattern ed ultime novità

Ugo Landini - Solution Architect

Giuseppe Bonocore - Solution Architect

#RedHatOSD

“With great power, there must also come–

great responsibility.”

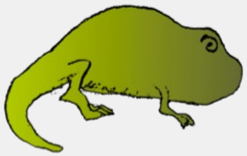
—SPIDERMAN (STAN LEE)

It is about the Journey

Re-Org to
DevOps

Self-Service,
On-Demand,
Elastic,
Infrastructure
as
Code
(Cloud)

Automation
Puppet, Chef,
Ansible
and/or
Kubernetes



CI & CD
Deployment
Pipeline

Advanced
Deployment
Techniques

Silicon
Valley
DotCom
Startup

Microservices

15 MAY 2018

Microservices in Adopt?

Rebecca Parsons
Chief Technology Officer

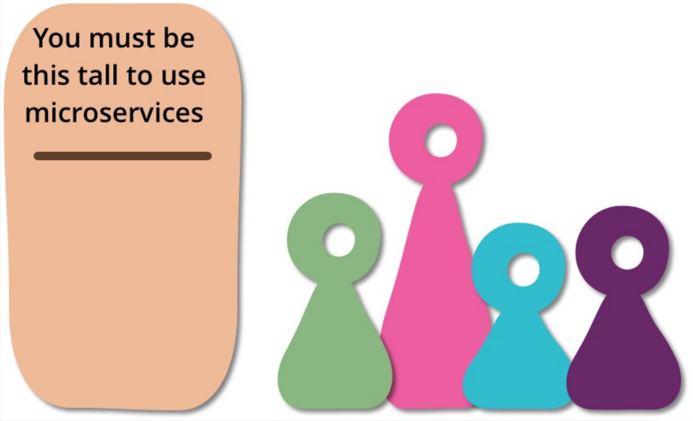
Technology » Microservices » Approaches »

70

Microservices play a major role in many organizations today. The movement gained momentum with the seminal article by James Lewis and Martin Fowler, followed by Sam Newman's book and numerous talks and articles by ThoughtWorkers, folks from Netflix, Google, and many others. Microservices quickly made it to the Trial ring on the

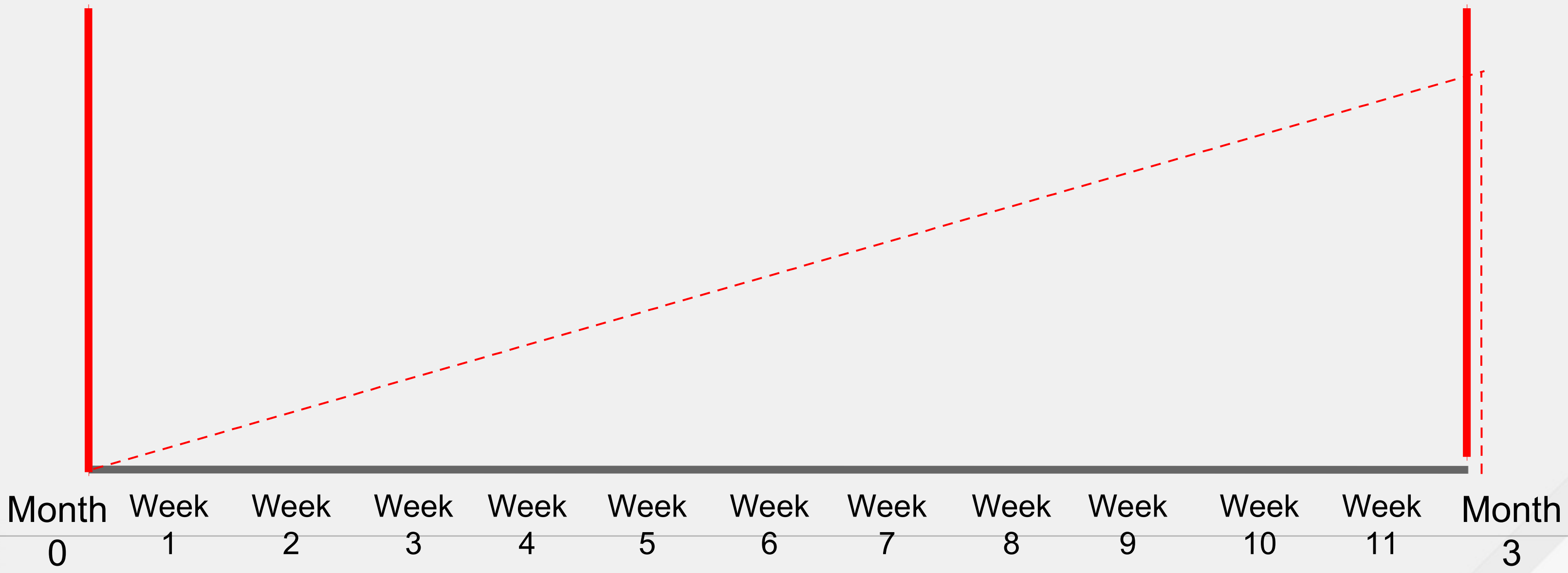
You Must Be This Tall

1. Self-Service, on-demand, elastic infrastructure as code
   (how many days/weeks to provision a new VM?)
2. Dev vs Ops
   (who is on the pager for production app outage?)
3. Automation
   (phoenix vs snowflake?)
4. CI & CD
5. Deployment Pipeline

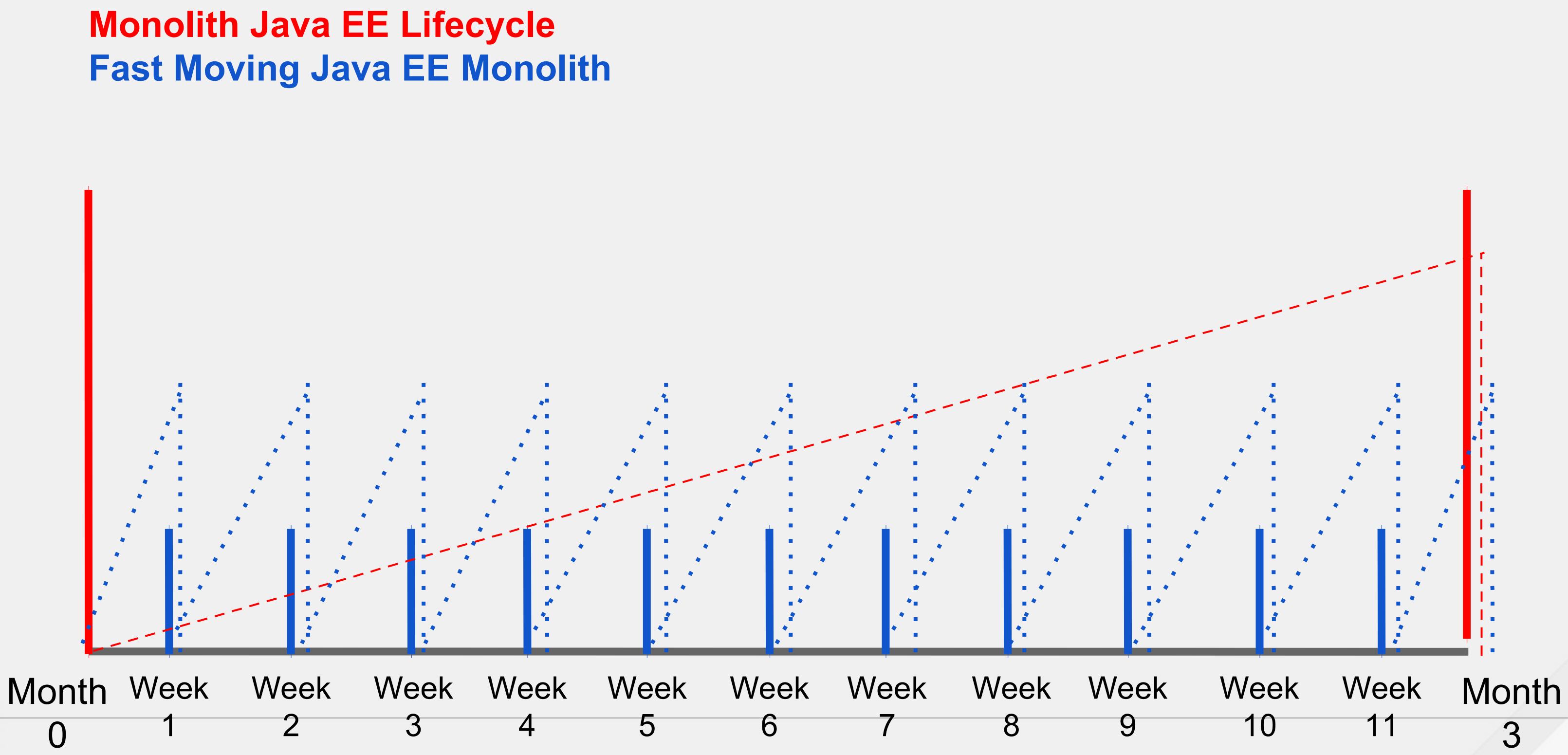


http://martinfowler.com/bliki/MicroservicePrerequisites.html

Maturing The Application Lifecycle



Maturing The Application Lifecycle

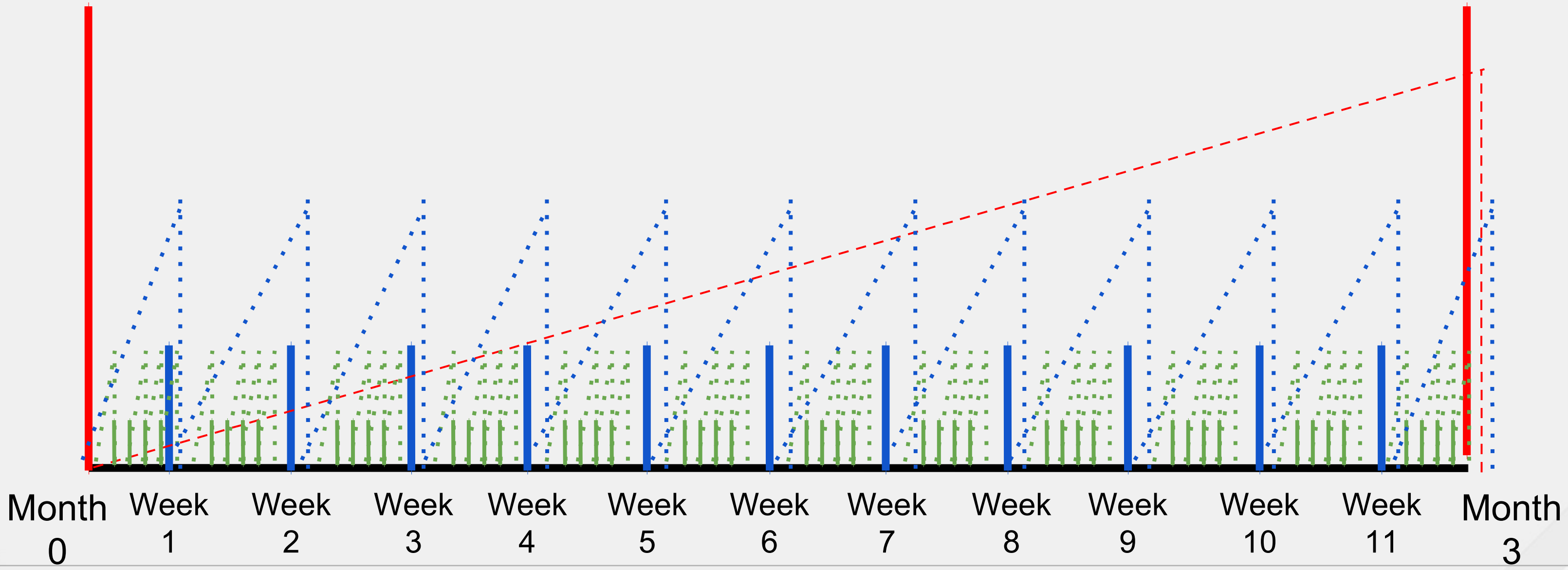


Maturing The Application Lifecycle

Monolith Lifecycle

Fast Moving Java EE Monolith

Java EE Microservices



Why Monolith To Microservices

Break things down (organizations, teams, IT systems, etc) down into smaller pieces for greater parallelization and autonomy and focus on reducing time to value.

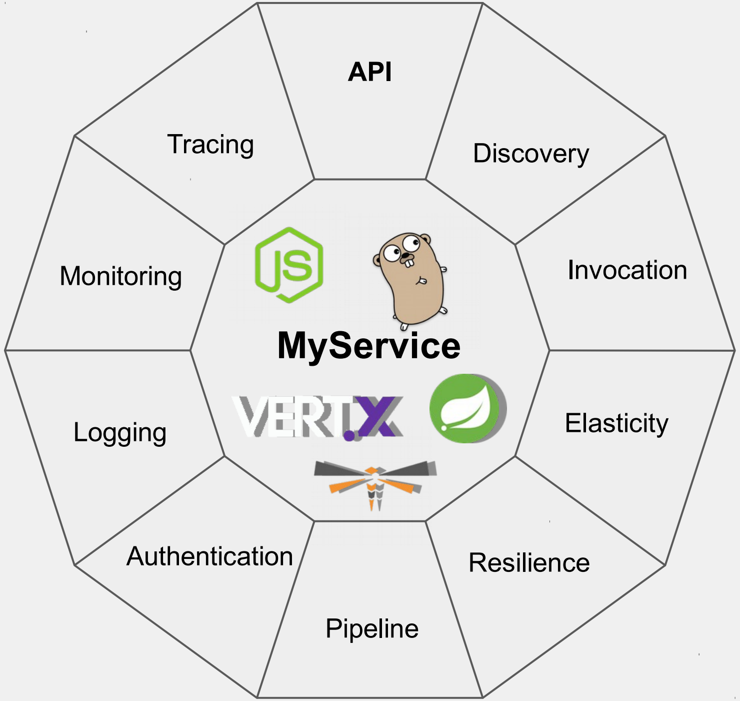
What’s the difference?

- Same ideas, new technologies (which will evolve in the future)
- But now, we should be able to bring a new feature in **production** in a few minutes

Microservice Principles/Characteristics

1. Deployment **Independence** - updates to an individual microservice have no negative impact to any other component of the system. Optimized for **Replacement**
2. Organized around **business** capabilities
3. **Products** not Projects
4. **API** Focused
5. **Smart** endpoints and dumb pipes
6. Decentralized Governance
7. Decentralized Data Management
8. Infrastructure Automation (infrastructure as code)
9. Design for failure
10. Evolutionary Design

Microservices'ilities

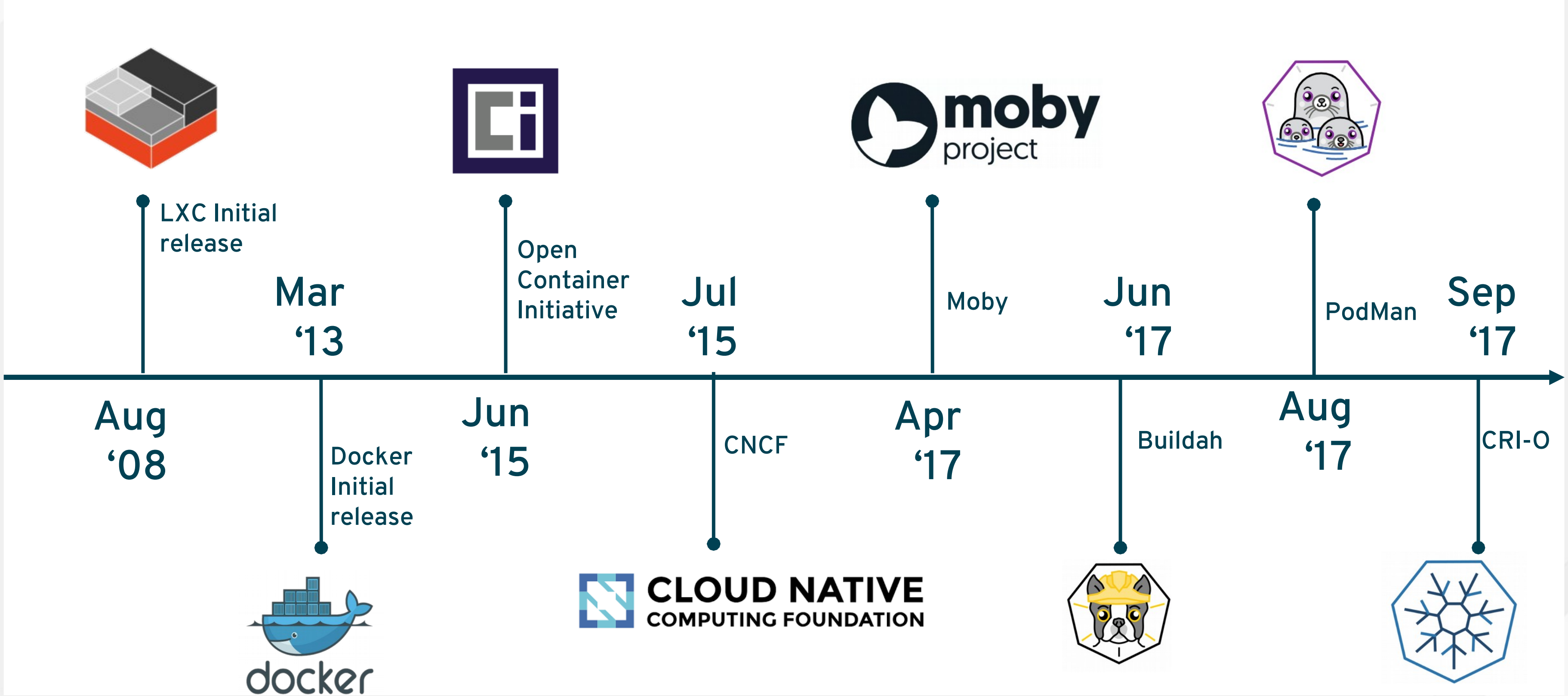


Microservice ≅ Container

https://www.flickr.com/photos/63975655@N07/

DOCCHER!

Container (no more) = Docker



Aug '08
LXC Initial release

Mar '13
Docker Initial release

Jun '15
Open Container Initiative

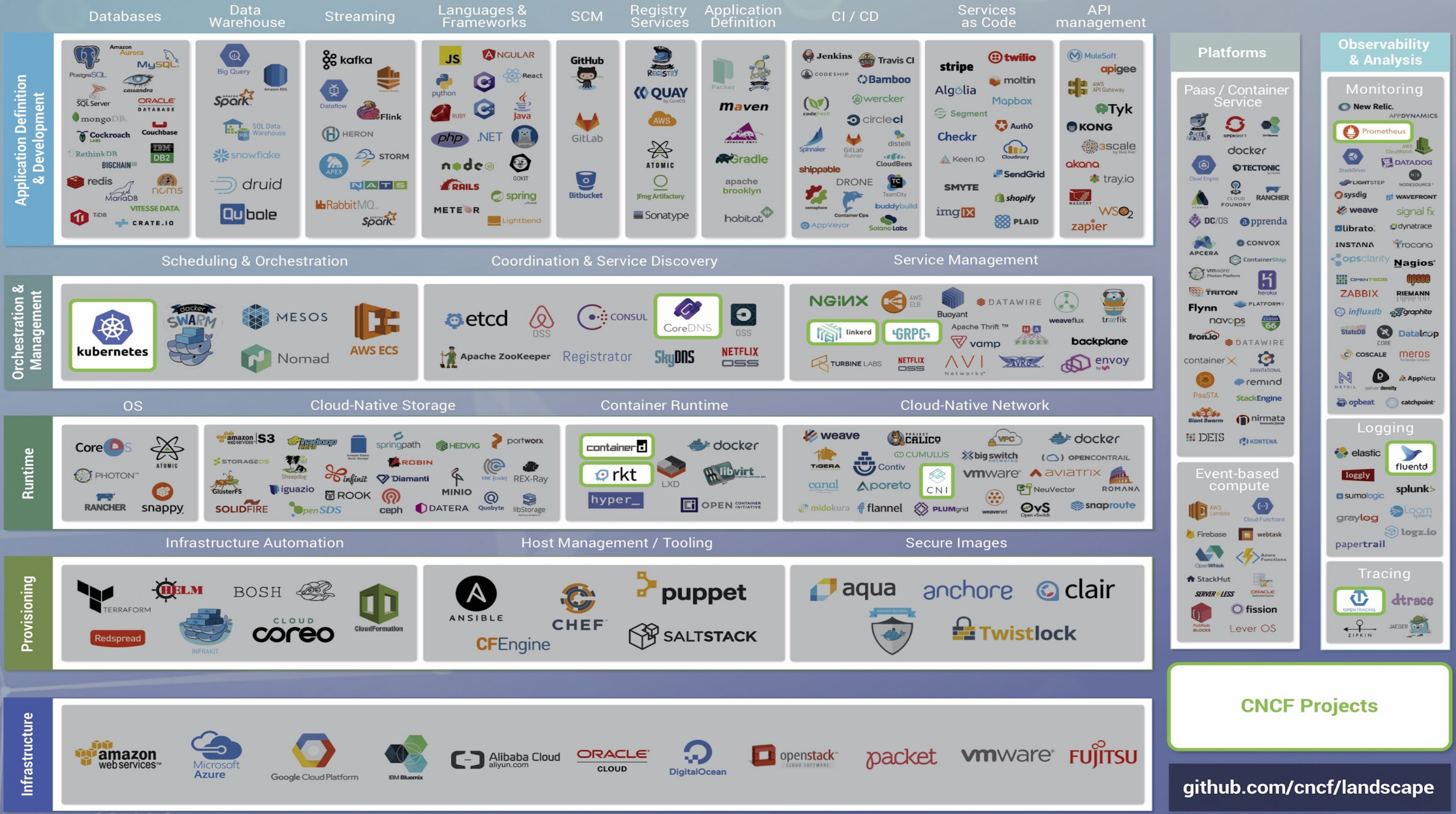
Jul '15
CNCF

Apr '17
Moby

Jun '17
Buildah

Aug '17
PodMan

Sep '17
CRI-O



Application Definition & Development

Databases | Data Warehouse | Streaming | Languages & Frameworks | SCM | Registry Services | Application Definition | CI / CD | Services as Code | API management

Orchestration & Management

Scheduling & Orchestration | Coordination & Service Discovery | Service Management

Runtime

OS | Cloud-Native Storage | Container Runtime | Cloud-Native Network

Provisioning

Infrastructure Automation | Host Management / Tooling | Secure Images

Infrastructure

Platforms

Paas / Container Service

Event-based compute

Observability & Analysis

Monitoring

Logging

Tracing

CNCF Projects

github.com/cncf/landscape

THE CLOUD-NATIVE APP DEV CHALLENGE

Databases
Data Warehouse
Streaming
Languages & Frameworks
SCM
Registry Services
Application Definition
CI / CD
Services as Code
API management
Application Definition & Development
Platforms
Paas / Container Service
Observability & Analysis
Monitoring
Tracing
Infrastructure Automation
Host Management / Tooling
Secure Images
Provisioning
Infrastructure
CNCF Projects
github.com/cncf/landscape

Config and Setup

24%

Of time spent building and maintaining developer environments.

Dev Tool Integration

41%

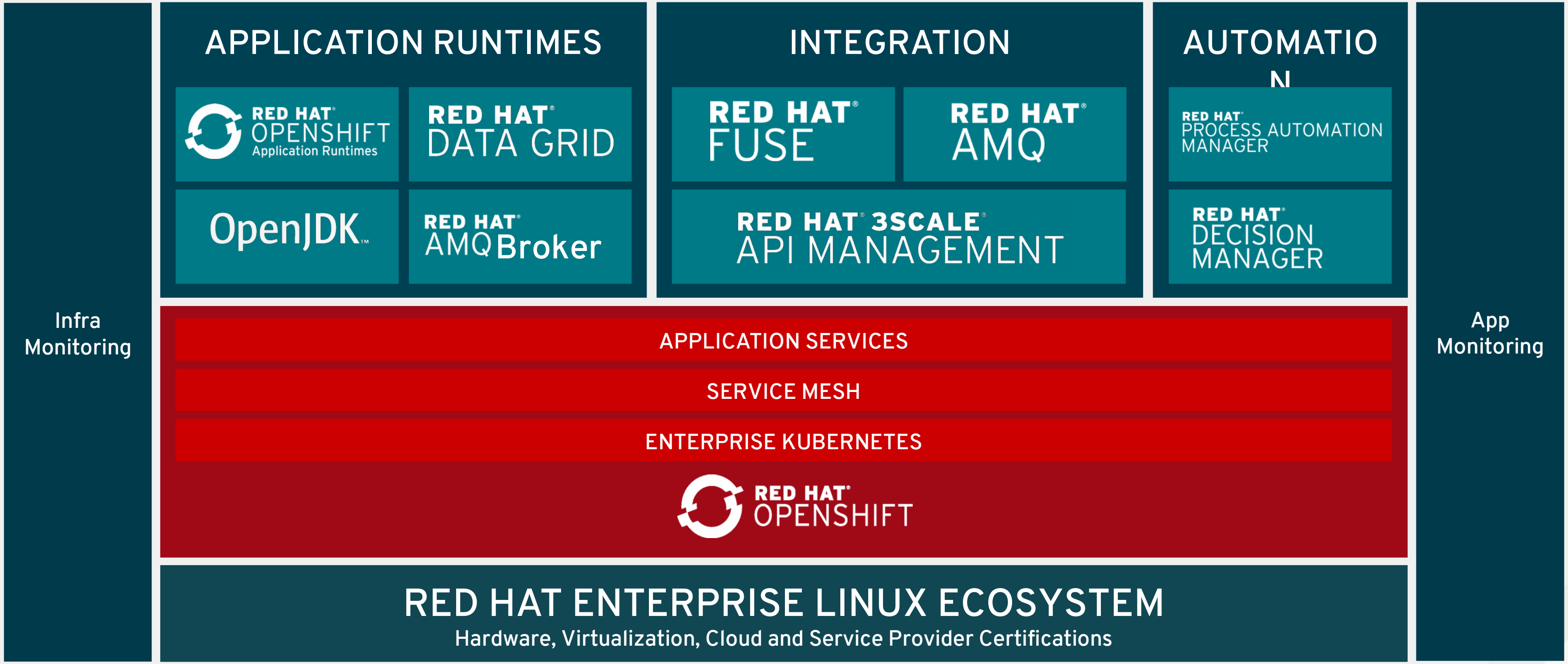
Of enterprises see non-integrated tools as an inhibitor to container adoption.

Source: Cloud Development Survey 2017 - Evans Data Corp

RED HAT® OPENSHIFT Application Runtimes

Modern, cloud-native application runtimes and an opinionated developer experience for organizations that are moving beyond 3-tier architectures and embracing cloud-native application development.

Red Hat Hybrid Cloud Development Platform



Infra Monitoring

APPLICATION RUNTIMES

RED HAT OPENSHIFT Application Runtimes

RED HAT DATA GRID

OpenJDK

RED HAT AMQ Broker

INTEGRATION

RED HAT FUSE

RED HAT AMQ

RED HAT 3SCALE API MANAGEMENT

AUTOMATIO N

RED HAT PROCESS AUTOMATION MANAGER

RED HAT DECISION MANAGER

App Monitoring

APPLICATION SERVICES

SERVICE MESH

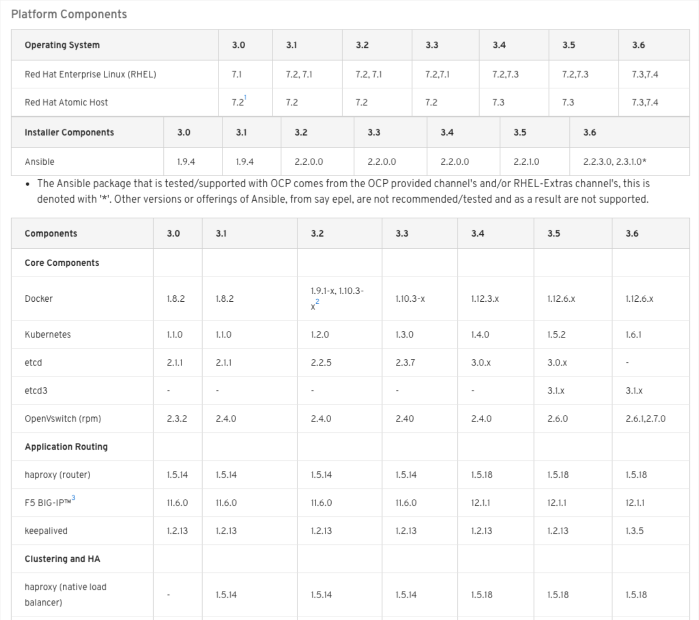
ENTERPRISE KUBERNETES

RED HAT OPENSHIFT

RED HAT ENTERPRISE LINUX ECOSYSTEM

Hardware, Virtualization, Cloud and Service Provider Certifications

Openshift Tested Integration



Platform Components

Operating System	3.0	3.1	3.2	3.3	3.4	3.5	3.6
Red Hat Enterprise Linux (RHEL)	7.1	7.2, 7.1	7.2, 7.1	7.2,7.1	7.2,7.3	7.2,7.3	7.3,7.4
Red Hat Atomic Host	7.2[1]	7.2	7.2	7.2	7.3	7.3	7.3,7.4

Installer Components	3.0	3.1	3.2	3.3	3.4	3.5	3.6
Ansible	1.9.4	1.9.4	2.2.0.0	2.2.0.0	2.2.0.0	2.2.1.0	2.2.3.0, 2.3.1.0*

- The Ansible package that is tested/supported with OCP comes from the OCP provided channel's and/or RHEL-Extras channel's, this is denoted with '*'. Other versions or offerings of Ansible, from say epel, are not recommended/tested and as a result are not supported.

Components	3.0	3.1	3.2	3.3	3.4	3.5	3.6
Core Components							
Docker	1.8.2	1.8.2	1.9.1-x, 1.10.3-x[2]	1.10.3-x	1.12.3.x	1.12.6.x	1.12.6.x
Kubernetes	1.1.0	1.1.0	1.2.0	1.3.0	1.4.0	1.5.2	1.6.1
etcd	2.1.1	2.1.1	2.2.5	2.3.7	3.0.x	3.0.x	-
etcd3	-	-	-	-	-	3.1.x	3.1.x
OpenVswitch (rpm)	2.3.2	2.4.0	2.4.0	2.40	2.4.0	2.6.0	2.6.1,2.7.0
Application Routing							
haproxy (router)	1.5.14	1.5.14	1.5.14	1.5.14	1.5.18	1.5.18	1.5.18
F5 BIG-IP™[3]	11.6.0	11.6.0	11.6.0	11.6.0	12.1.1	12.1.1	12.1.1
keepalived	1.2.13	1.2.13	1.2.13	1.2.13	1.2.13	1.2.13	1.3.5
Clustering and HA							
haproxy (native load balancer)	-	1.5.14	1.5.14	1.5.14	1.5.18	1.5.18	1.5.18

- 100+ defects fixed between every upstream Kubernetes and commercial OpenShift release
- 140+ combinations of common products tested with every *minor* OpenShift release, incl. Storage drivers, networking, database images, …
- Tested for performance & scalability, security and reliability

https://access.redhat.com/articles/2176281

Ok, so it’s (also) about being lighter?

Theoretically, yes. But, beware:

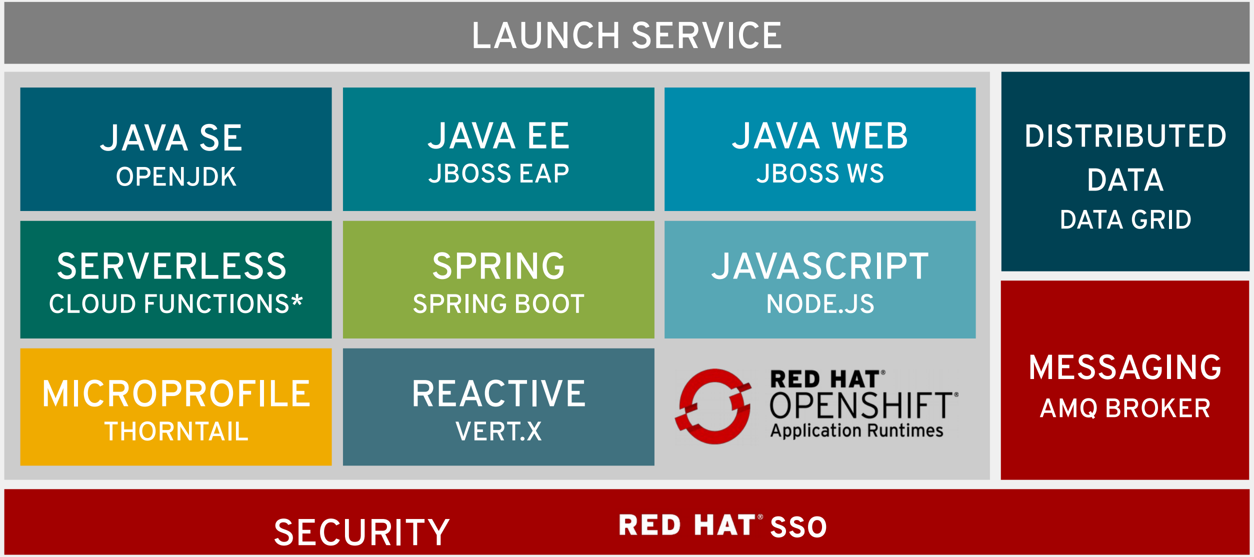
- A simple ReST service deployed in EAP used ⅕ of the memory used by Spring Boot under load and was **2x** faster!

Runtime (framework)	Boot time server only	Boot time including app deployment	Memory usage without load	Memory usage under load	Measured throughput
JBoss EAP (Java EE)	2 - 3 sec	**3 sec**	40 MB	**200 - 400 MB**	23K req/sec
JBoss EAP (Spring)	2 - 3 sec	7 sec	40 MB	500 - 700 MB	9K req/sec
JBoss WS/Tomcat (Spring)	**0 - 1 sec**	8 sec	40 MB	0.5 - 1.5 GB	8K req/sec
Fat JAR (Spring Boot)	N/A	**3 sec**	**30 MB**	0.5 - 2.0 GB	11K req/sec
Fat JAR (Thorntail)	1-2 sec	5 sec	**30 MB**	250 - 350 MB	**27K req/sec**

Decision Points For Selecting The Runtimes

Selection Consideration	Project Type				Framework Pref		Learning Effort			Deployment Pkg		
Runtimes	Cloud Native (new)	Cloud Enable (existing)			Java EE	Non-Java EE	No	Little	Invest	Thin	Fat	Hollow
		Lift & Shift	Connect & Enhance	Refactor & Rewrite								
EAP	+	+	+	+	+		+			+		
Thorntail	+		+	+	+	+	+	+		+	+	+
Vert.x	+		+	+		+			+	+	+	
Node.js	+		+	+		+			+	+		
Tomcat	+	Spring Boot	+	+		+	+			+	Spring Boot	

Application Runtimes



LAUNCH SERVICE

JAVA SE
OPENJDK

JAVA EE
JBOSS EAP

JAVA WEB
JBOSS WS

DISTRIBUTED DATA
DATA GRID

SERVERLESS
CLOUD FUNCTIONS*

SPRING
SPRING BOOT

JAVASCRIPT
NODE.JS

MICROPROFILE
THORNTAIL

REACTIVE
VERT.X

RED HAT® OPENSHIFT®
Application Runtimes

MESSAGING
AMQ BROKER

SECURITY RED HAT® SSO

Pre-configured Missions and Boosters
Integration with RH Developer, CI/CD tools, Security Services
Optimized for OpenShift / Kubernetes Services
Available Application Migration Toolkit
Python, Go and .Net also supported by Red Hat (with a different SLA)

Facilitate cloud native app development ON THE HYBRID CLOUD:

✓ Faster getting started

✓ Simplify container dev

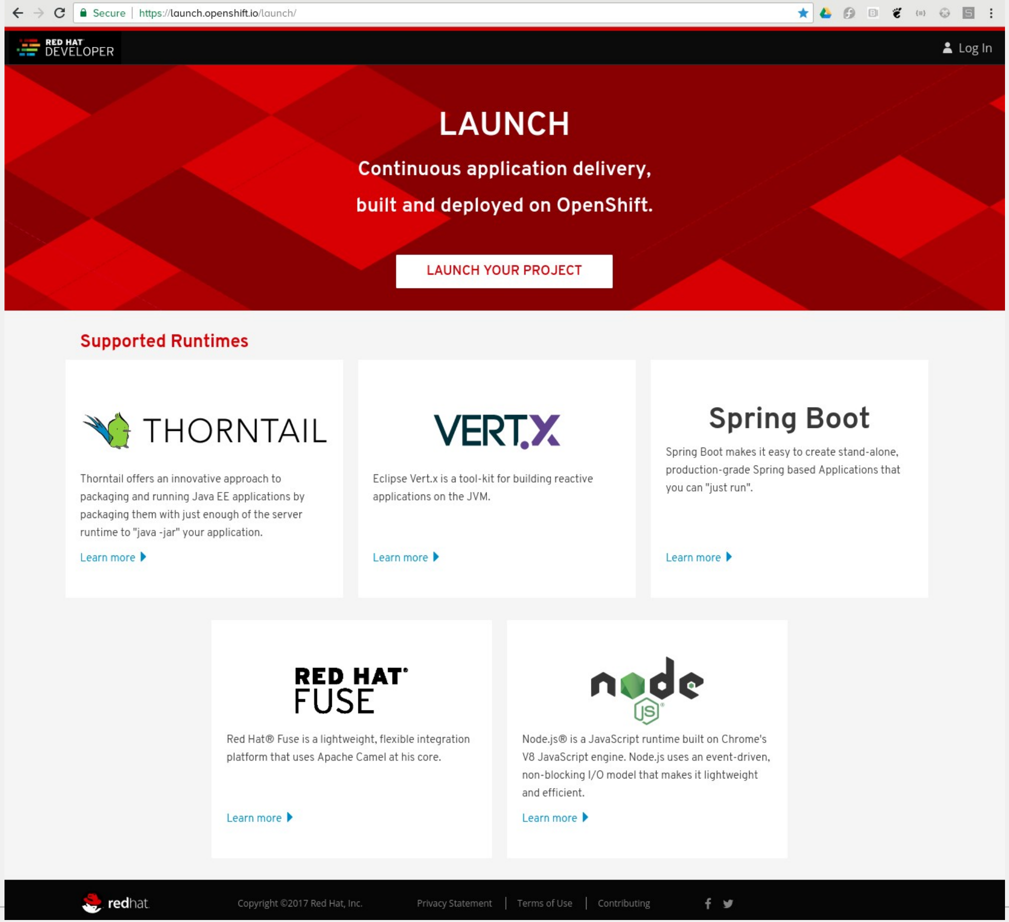
✓ Automate DevOps

✓ Standardize tools/processes

✓ Fully supported JDK

*Coming Soon

launch.openshift.io



Secure | https://launch.openshift.io/launch/

RED HAT DEVELOPER

Log In

LAUNCH

Continuous application delivery,
built and deployed on OpenShift.

LAUNCH YOUR PROJECT

Supported Runtimes

THORNTAIL

Thorntail offers an innovative approach to packaging and running Java EE applications by packaging them with just enough of the server runtime to "java -jar" your application.

Learn more

VERT.X

Eclipse Vert.x is a tool-kit for building reactive applications on the JVM.

Learn more

Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

Learn more

RED HAT® FUSE

Red Hat® Fuse is a lightweight, flexible integration platform that uses Apache Camel at his core.

Learn more

node JS®

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Learn more

redhat

Copyright ©2017 Red Hat, Inc.

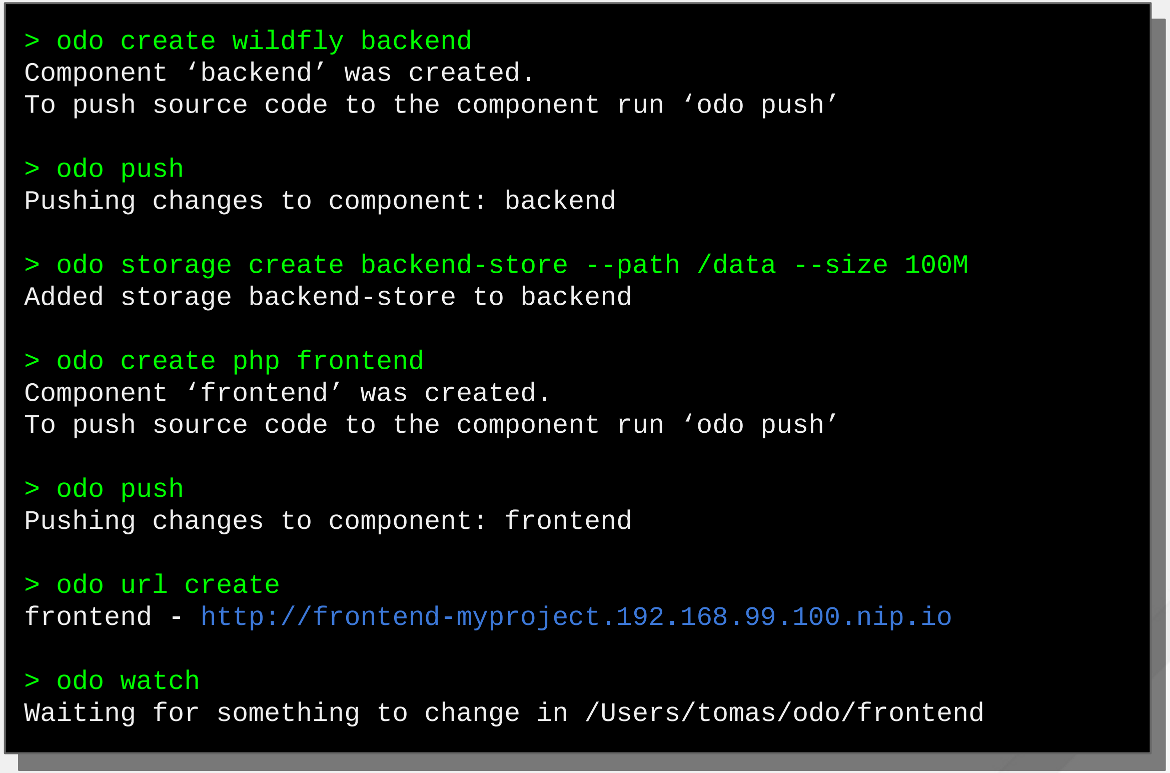
Privacy Statement | Terms of Use | Contributing

Openshift-do: A Cli For Developers

Openshift-DO (“odo”) is a new CLI plugin for OpenShift 3.9+ that is tailored for developer syntax and workflows.

Goal is to make it simple for a developer to create an app, add components (like a database) and expose it without needing to know Kubernetes.

In tech preview now.



```
> odo create wildfly backend
Component ‘backend’ was created.
To push source code to the component run ‘odo push’

> odo push
Pushing changes to component: backend

> odo storage create backend-store --path /data --size 100M
Added storage backend-store to backend

> odo create php frontend
Component ‘frontend’ was created.
To push source code to the component run ‘odo push’

> odo push
Pushing changes to component: frontend

> odo url create
frontend - http://frontend-myproject.192.168.99.100.nip.io

> odo watch
Waiting for something to change in /Users/tomas/odo/frontend
```

The books you'll need to read

The DevOps Handbook
How to Create World-Class Agility, Reliability, & Security in Technology Organizations
Gene Kim, Jez Humble
O'Reilly
Migrating to Microservice Databases
From Relational Monolith to Distributed Data
Compliments of Red Hat Developers
O'Reilly
Building Microservices
Designing Fine-Grained Systems
The Pragmatic Programmers
Release It! Second Edition
Design and Deploy Production-Ready Software
Michael T. Nygard
O'Reilly
Microservices for Java Developers
A Hands-On Introduction to Frameworks & Containers
Christian Posta
Compliments of Red Hat Developers
The Addison-Wesley Signature Series
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler
The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Revised with New Resource Guide
Gene Kim, Kevin Behr, and George Spafford
O'Reilly
Building Reactive Microservices in Java
Asynchronous and Event-Based Application Design
Clement Escoffier
Compliments of Red Hat Developer Program
Domain-Driven Design
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

redhat.

GRAZIE PER L’ATTENZIONE

Ugo Landini - Solution Architect

Giuseppe Bonocore - Solution Architect

#RedHatOSD