



RED HAT OPENSIFT APPLICATION RUNTIMES(RHOAR) 를 활용한 Cloud Native App 전환

2018. 02

JongJin Lim (jonlim@redhat.com)

Specialist Solution Architect, AppDev

Red Hat Korea

BY 2020 **75%** OF APPLICATION PURCHASES
SUPPORTING DIGITAL BUSINESS WILL BE “BUILD”
NOT “BUY”

GARTNER Forecast analysis <http://www.gartner.com/newsroom/id/3119717>

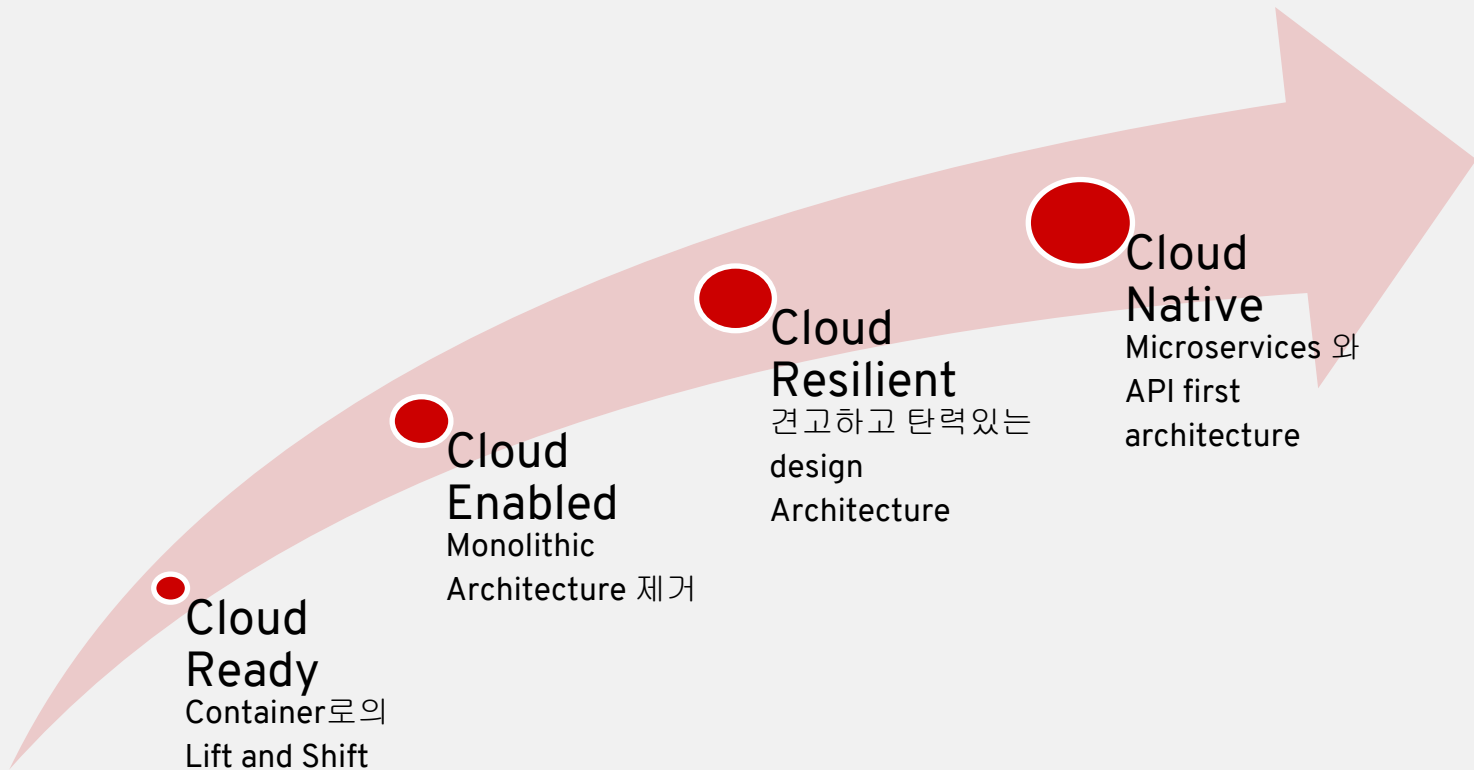
Innovation that creates
differentiation leading to
customer preference

BUILD

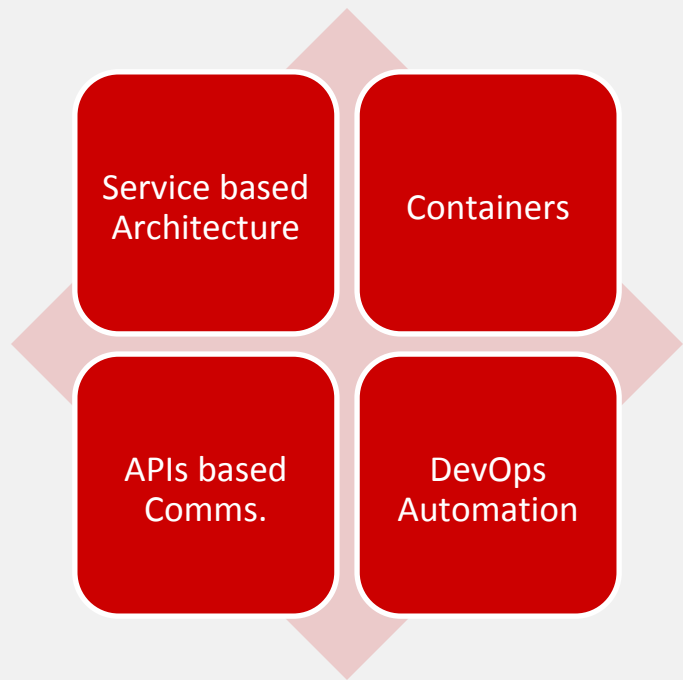
BUY

Everything else, not a source
of differentiation

CLOUD NATIVE RAMP UP MODEL



CLOUD NATIVE APPLICATION DEVELOPMENT



Cloud-native는 Application을 만들고 실행하는 방법

- Microservices 같은 modular, loose coupling 기반 서비스
- Containers 를 이용한 간소화된 배포 및 실행
- CI/CD, AutoScale 등 프로세스 자동화를 사용한 DevOps Automation 구현
- API 기반 통신 구현

Infrastructure(Private, Public, Hybrid Cloud)와는 관련 없는 Application 개발 방법론

TRANSFORM YOUR BUSINESS - CLOUD NATIVE WAY

Business

비즈니스 목표에 맞게
application 전략 조정

시장 변화에 신속하게 대응

새로운 비즈니스 기회 창출

Development

개발 프레임워크 및 플랫폼
선택

프로덕션 환경으로의 빠른 빌드
및 배포

자동화된 self-service
프로비저닝

Operations

DevOps의 표준화

Application 모니터링이 가능한
플랫폼 유지

Auto scaling이 가능한 HA구성

CLOUD NATIVE DEVELOPMENT의 가장 큰 변화

SPEED, RESILIENCY AND AGILITY

46배
많은
Deployment
주기

440배
Lead time
변화 시간
가속

96배
recovery
시간 가속

5배
낮은 Change
실패 확률

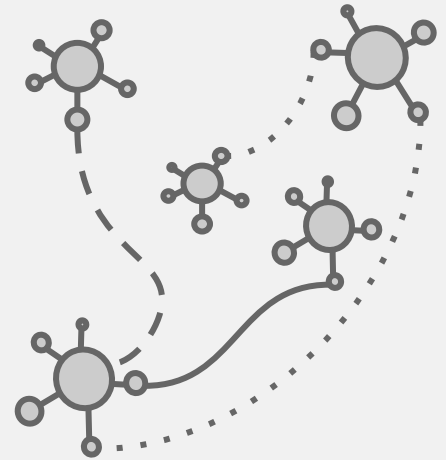
EVOLUTION OF MICROSERVICES

Trends : Microservices

“... is an approach to developing a single application as a suite of small services, each running in its **own process** and **communicating with lightweight** mechanisms, often an HTTP resource API. These services are built around business capabilities and **independently deployable** by **fully automated** deployment machinery. There is a bare minimum of **centralized management** of these services, which may be written in **different programming languages** and use different data storage technologies.”

- *Martin Fowler*

<http://martinfowler.com/articles/microservices.html>



Why are organization adopting microservices for **Cloud Native App**?

빠른 배포



간단한 코드



편리해진 개발



쉬운 확장



MICROSERVICES의 진화 (2014 - today)



Microservices 개발을 Infra와 별개의 Stack으로 분류

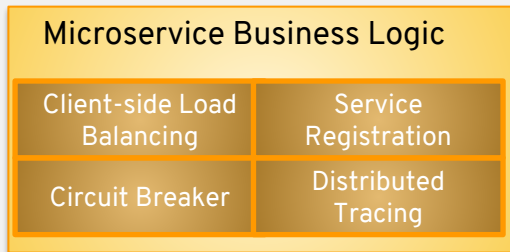


Infrastructure as a Service (IaaS)

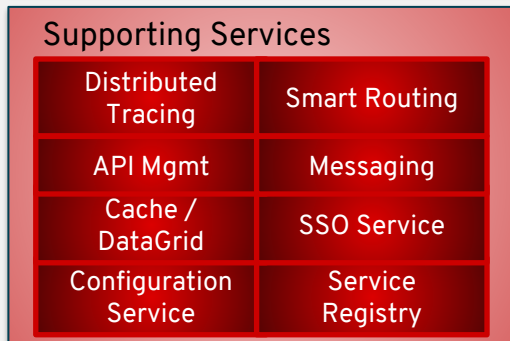


2014

MICROSERVICES의 진화 (2014 - today)



← - - - 인프라의 문제로 Business Logic의 복잡도 증가



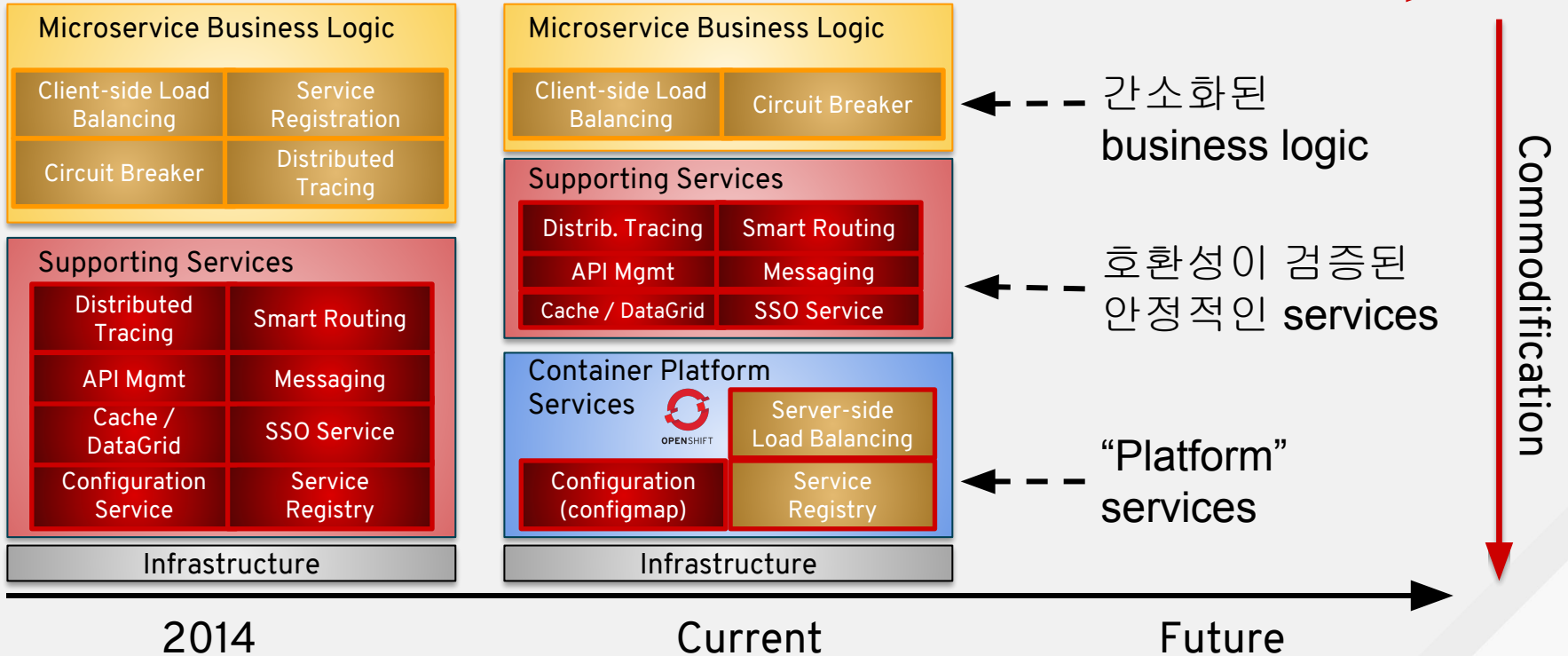
← - - - “복잡한 MS 관리를 위한” 다수/다종의 지원 서비스



← - - - Infrastructure as a Services (IaaS)

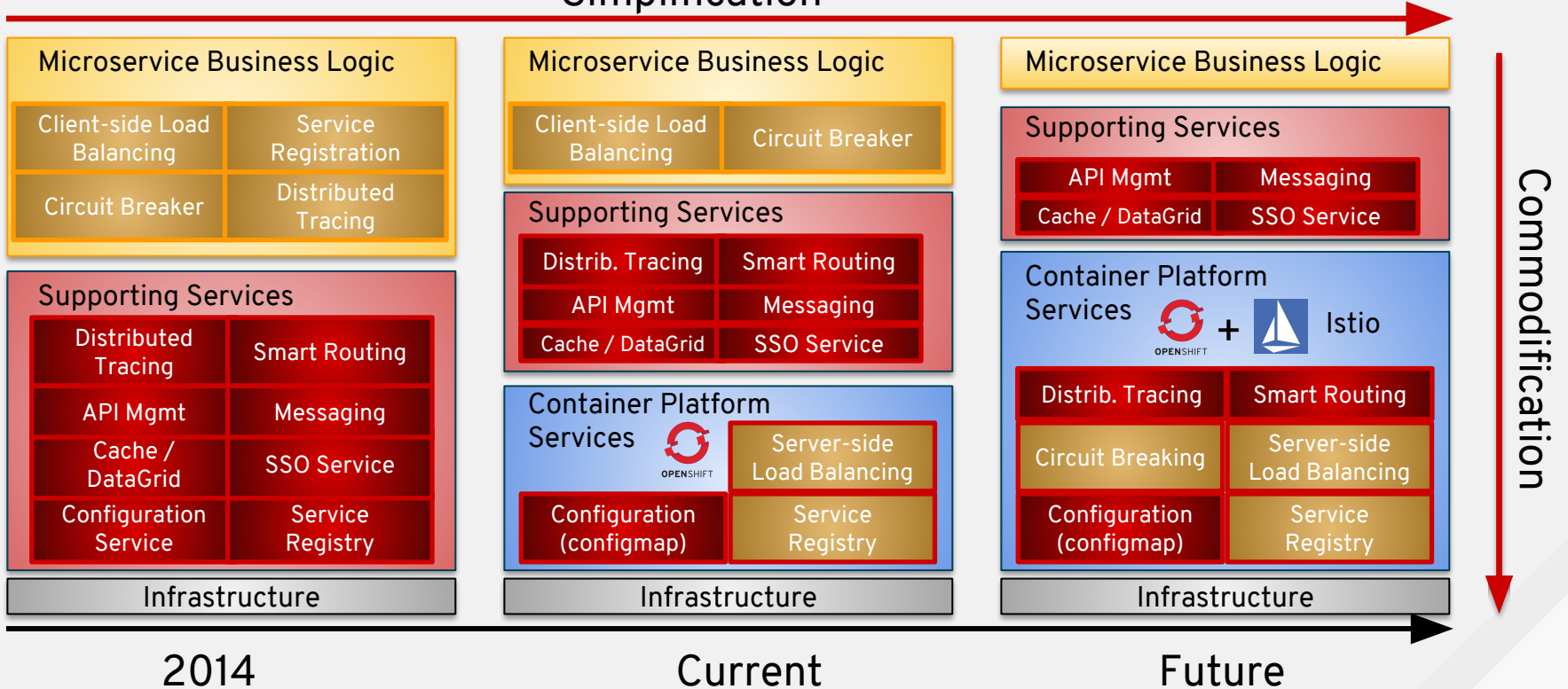
MICROSERVICES의 진화 (2014 - today)

Simplification



MICROSERVICES의 진화 (2014 - today)

Simplification



Red Hat Strategy

THE RED HAT PROVEN METHOD

Modernization 및 Migration으로 비즈니스 성장 촉진

APPLICATION MODERNIZATION & MIGRATION

MODERNIZATION

MIGRATION

BETTER
SOFTWARE
ARCHITECTURE

AGILE
INTEGRATION

STREAMLINE
APPLICATION
LIFECYCLE

CONTINUOUS
INNOVATION

APPLICATION
SERVERS

INTEGRATION
PLATFORMS

BUSINESS
DECISION
MANAGEMENT

APPLICATION
INFRASTRUCTURE

비용을 넘어선 고객의 가치 - Digital transformation



자원의 효율적인 배분



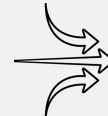
복잡성 감소 및 효율성 증가



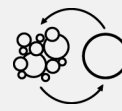
VENDOR LOCK-IN 회피 및
융통성 있는 라이선스 모델



속도 및 생산성 증가



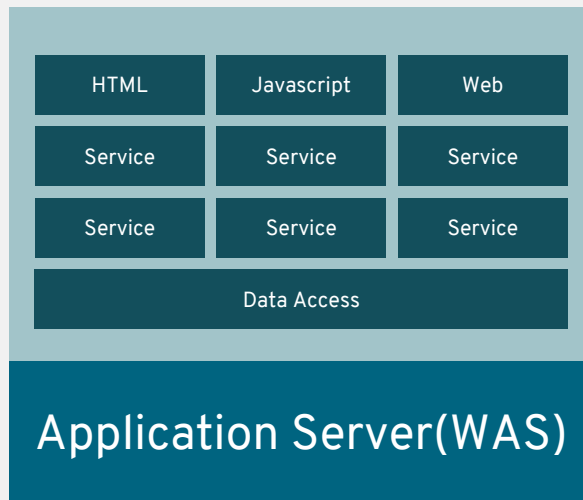
기술적인 이슈 해결 및
기술력 내재화



AGILE / DEVOPS 적용

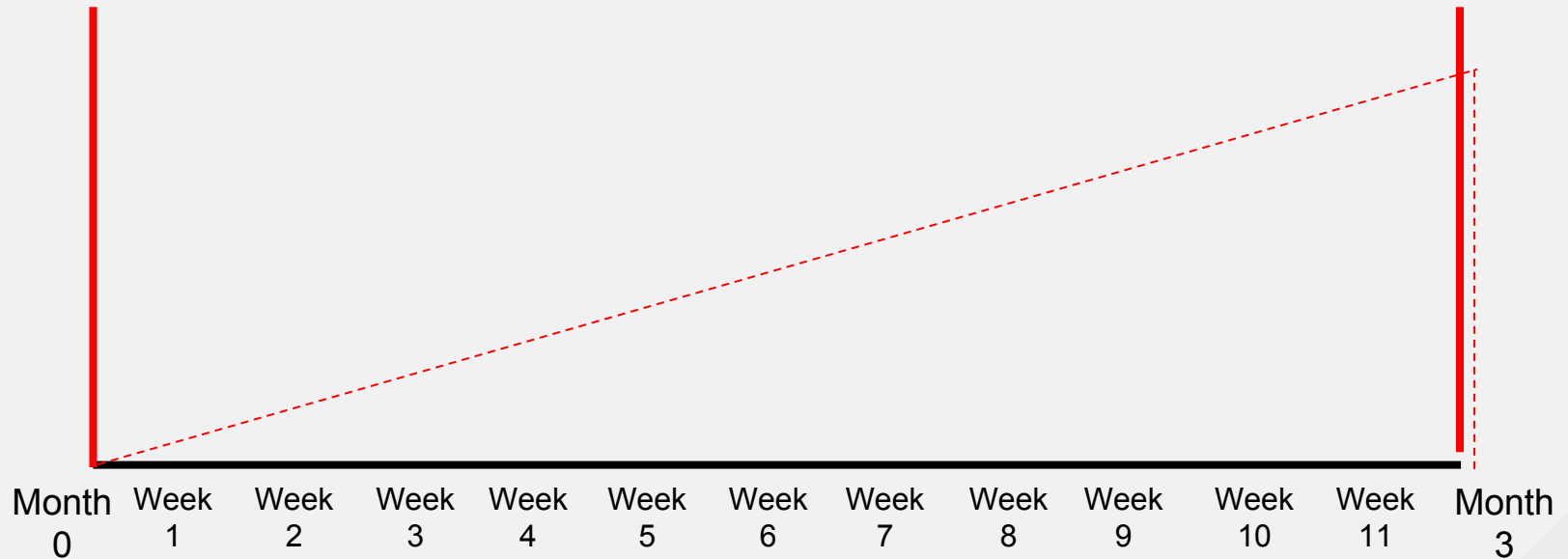
APPLICATIONS TODAY

Application Server(WAS)에 Monolithic Application 배포

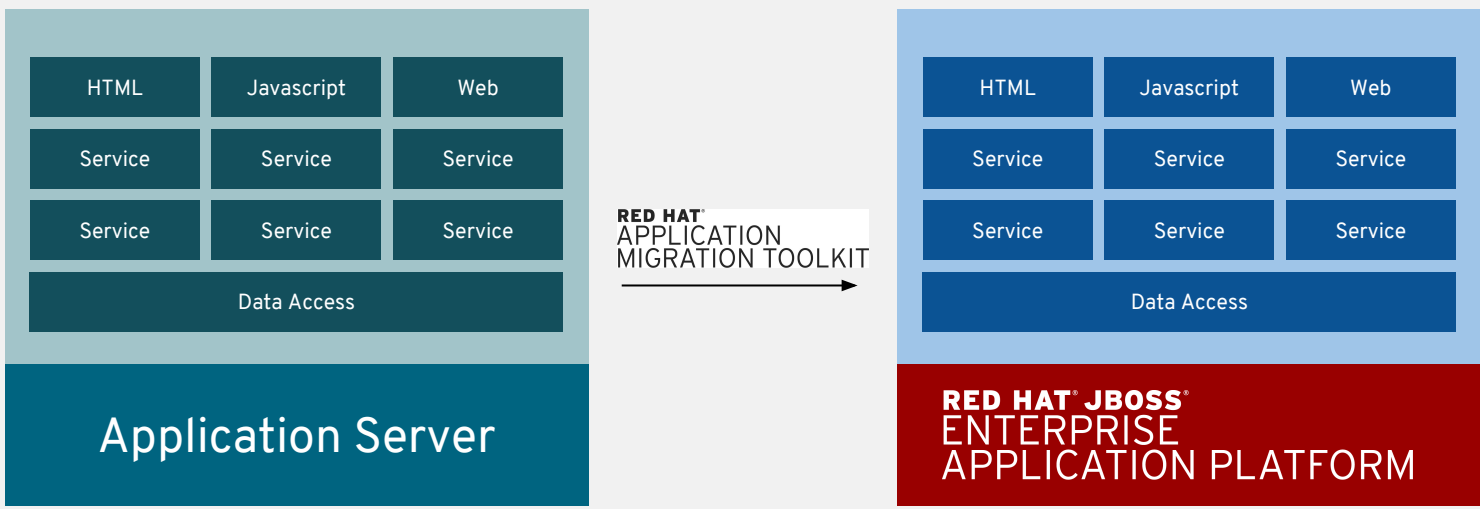


MATURING THE APPLICATION LIFECYCLE

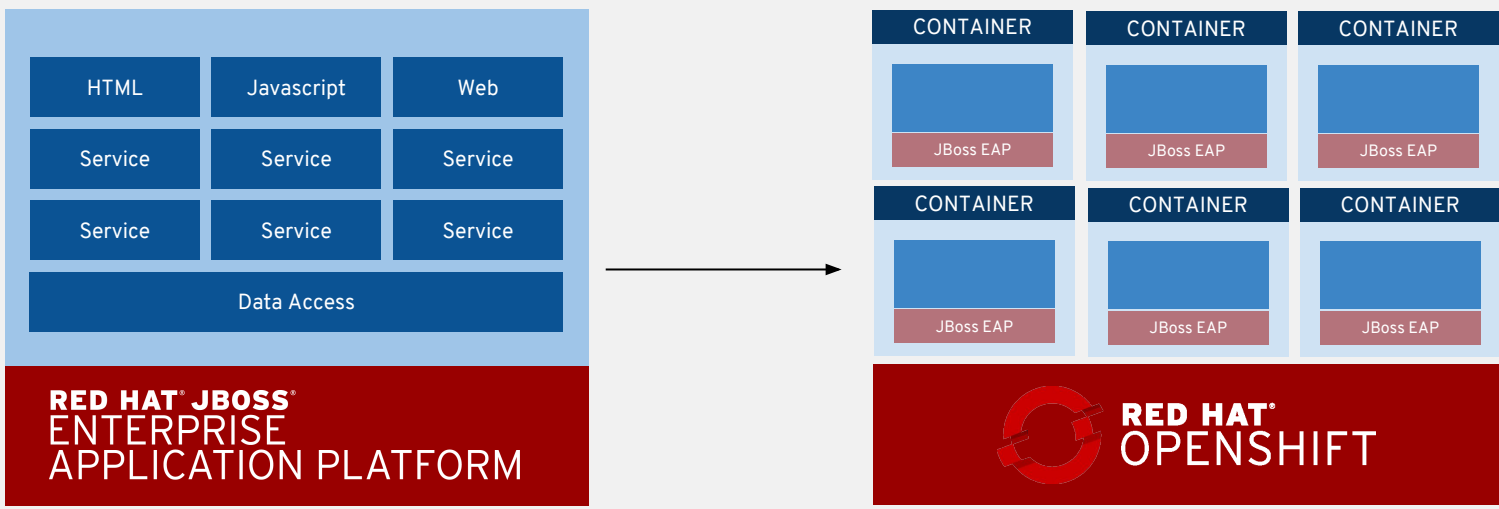
Monolithic Java EE Application Lifecycle



MONOLITH에서 CLOUD로의 LIFT-AND-SHIFT



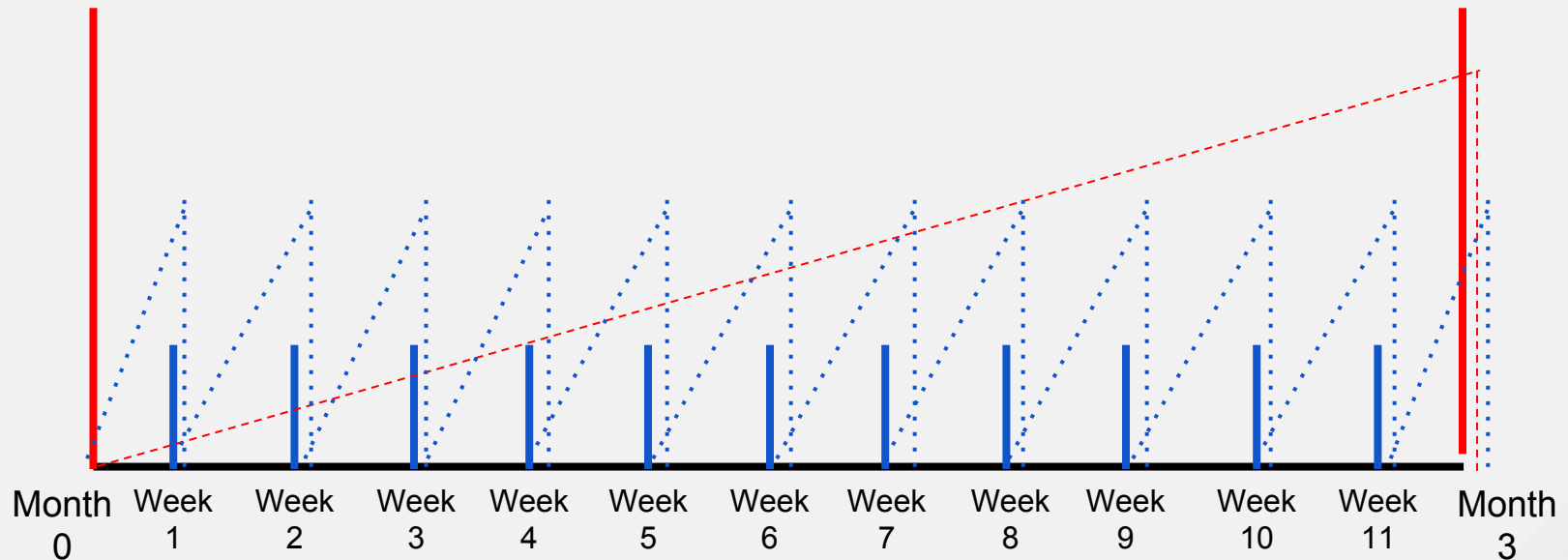
MONOLITH에서 CLOUD로의 LIFT-AND-SHIFT



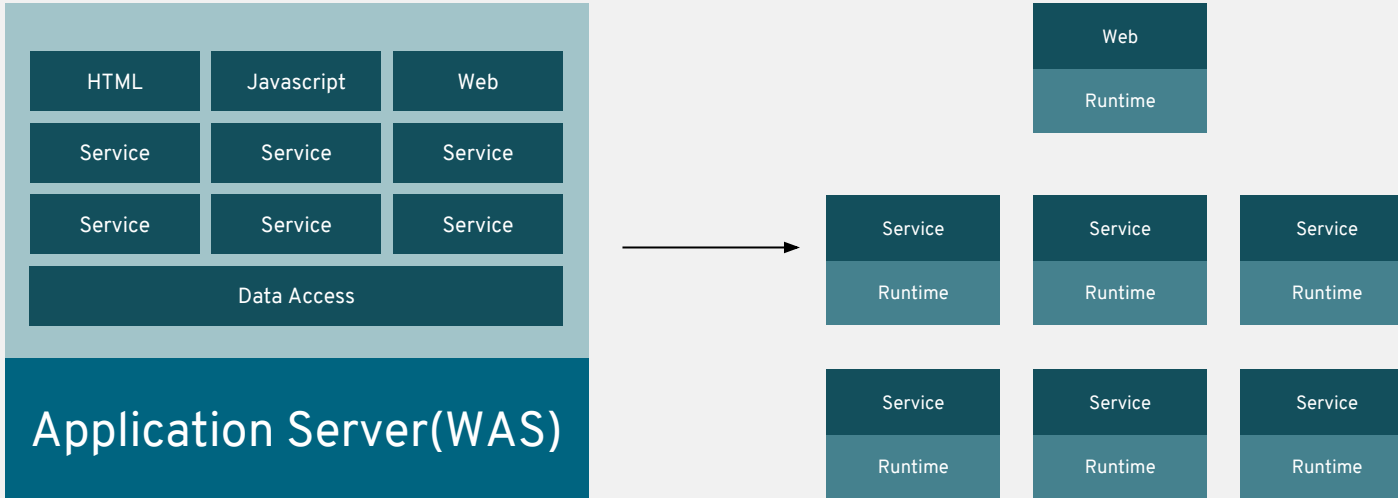
MATURING THE APPLICATION LIFECYCLE

Monolith Java EE Lifecycle

Fast Moving Java EE Monolith



MICROSERVICES



WHY MONOLITH TO MICROSERVICES

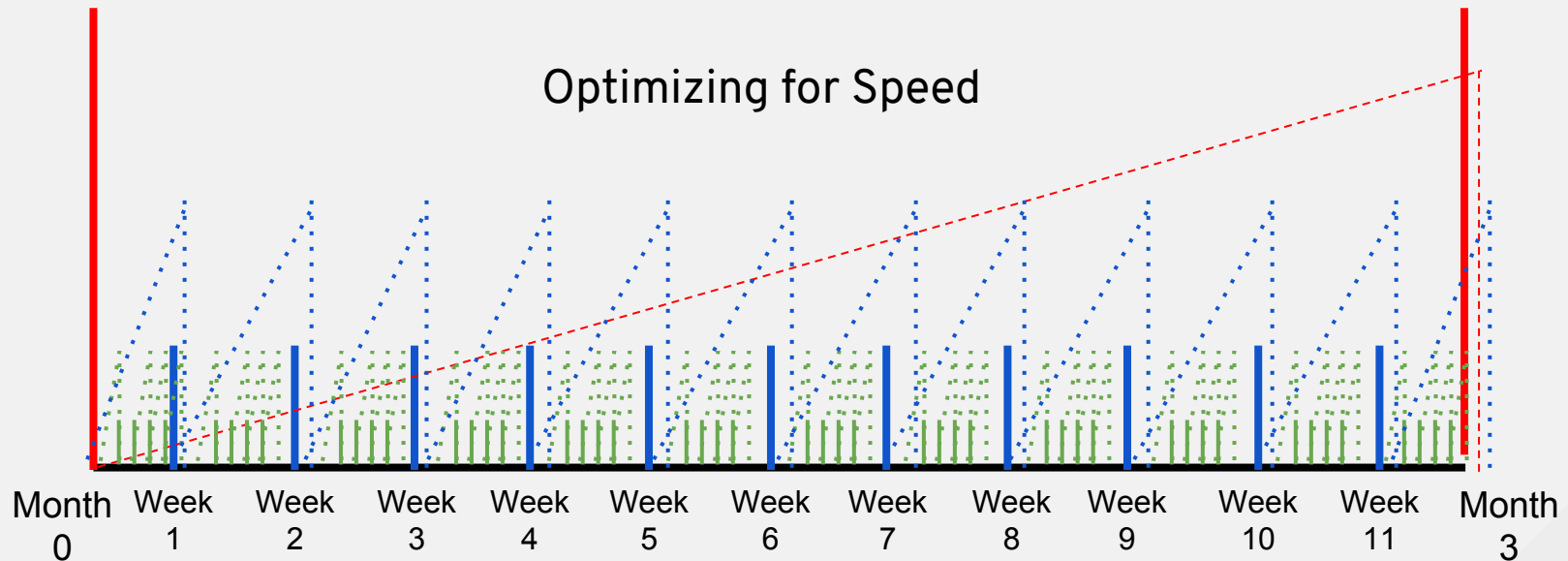
Break things down (organizations, teams, IT systems, etc) down into **smaller pieces** for **greater parallelization and autonomy** and focus on **reducing time to value**.

MATURING THE APPLICATION LIFECYCLE

Monolith Lifecycle

Fast Moving Java EE Monolith

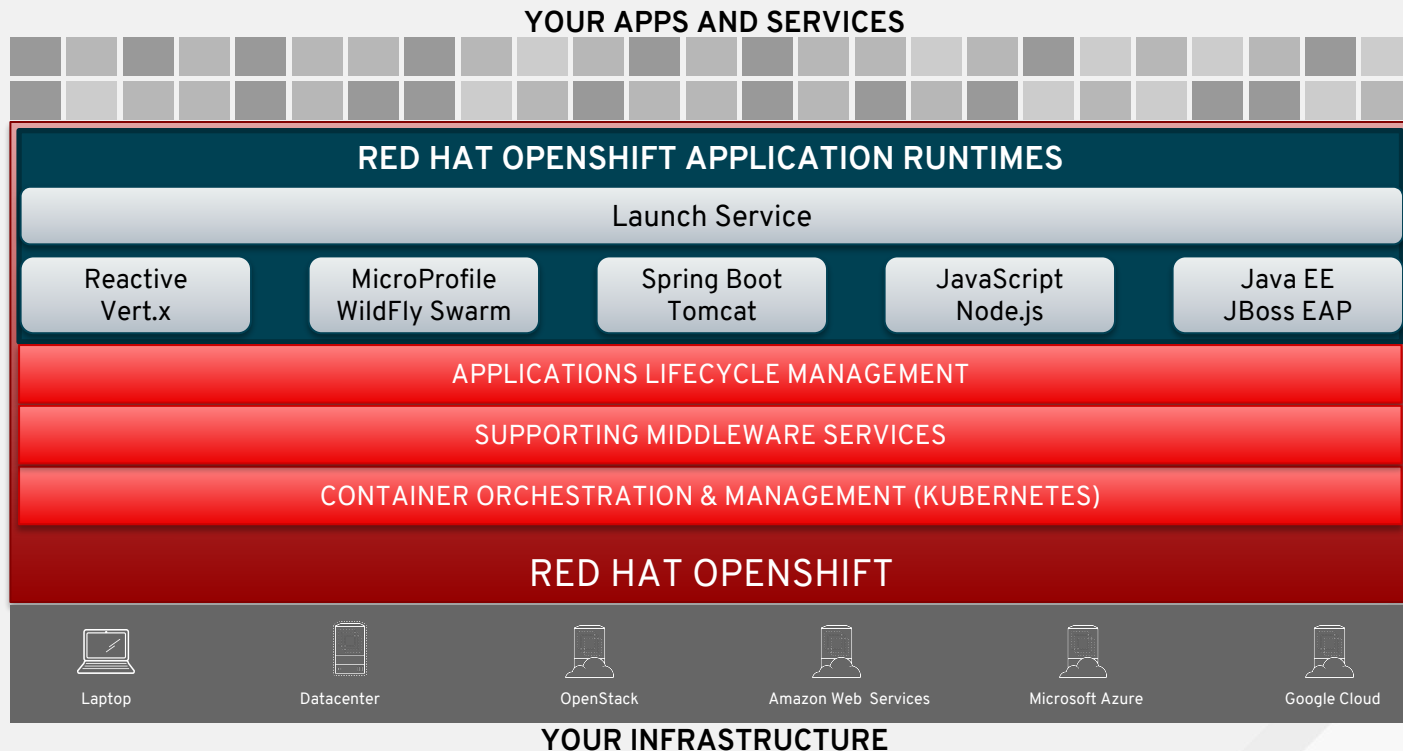
Java EE Microservices from RHOAR.



RED HAT OPENSIFT APPLICATION RUNTIMES

Cloud Native Application 개발을 위한 표준 런타임 및 프레임워크 세트 제공

- ✓ 간단한 개발
- ✓ 유연성 강화
- ✓ DevOps 자동화



SPEED UP APPLICATION DEVELOPMENT AND DELIVERY

RHOAR의 Pain-features-benefits

MSA개발의 어려움
(ARCHITECTURE)

- OpenShift와 Middleware 서비스들의 통합
- 빠른 준비가 가능한 Launch Service
- Application Missions and Boosters

간단한 개발

Lock-in에 대한 우려
(PLATFORM)

- Multi-Cloud 배포
- Multi-Runtimes 및 Frameworks 제공
- 다양한 app 적용 가능

유연성 강화

사일로 개발, 테스트,
배포 및 폭포수 개발
프로세스
(PROCESS)

- CI/CD tool과의 통합 (Git, Jenkins, Maven 등)
- DevOps pipeline 자동화 (A/B, Canary 배포 등)
- Openshift.io 개발 tool 연동


DevOps
자동화

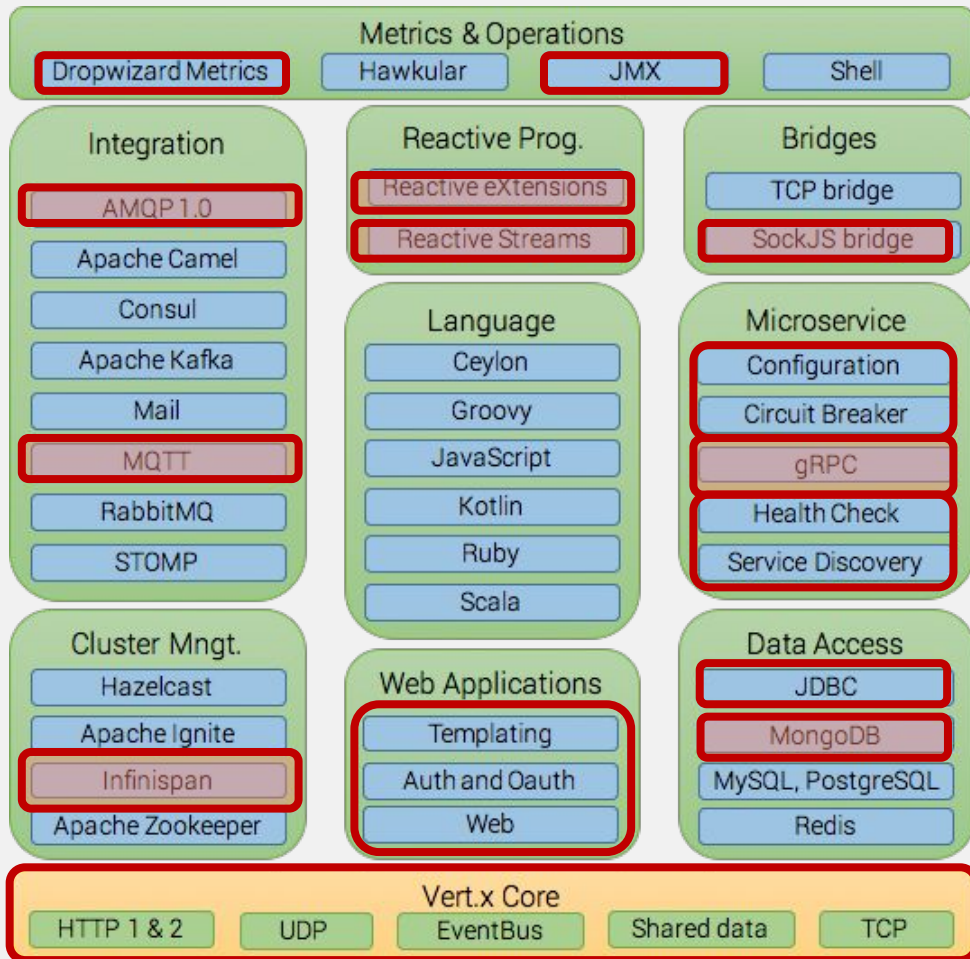
VERT.X

비동기식 **Non-blocking** 모델을 사용하여
JVM 상에서 분산 및 리액티브
Application을 구현하는 Toolkit

- Polyglot [Java supported]
- Integratable
- Embeddable
- Pragmatic
- Freedom

 Productized

 Technical Preview



Build microservices

- Embeddable (Fat Jar)
- Lightweight
- Modular & extensible
- Built from WildFly
(Trusted and Reliable)



Upstream (Unsupported)

Flyway

JMS

Jolokia

Logstash

Vert.x Integration

Infinispan

Fluentd

Consul

jGroups

Swagger

Spring

Certifications

Hystrix

Ribbon

MySQL

Oracle DB

Additional Supported Fractions

Metrics

Health

Configuration

Monitor

Keycloak

Topology

Supported Specifications

Java EE 7 Web Profile*

MicroProfile 1.0

Node.js (Tech Preview)

Server-side JavaScript runtime

Benefits

- JavaScript 기반으로 용이성
- Event driven with NIO
- 개발 생산성
- Performance and scalability

Open Source

크고, 활기찬 Community



~~150133~~ 174,235

total packages



~~66873631~~ 93,119,139

downloads in the last day



~~355187738~~ 484,835,350

downloads in the last week



~~1477311790~~ 2,027,961,536

downloads in the last month



Microservices 개발을 위한 Spring Framework

Spring (MicroProfile, Java EE, Vert.x, ...) app을 기동하기 위한 기반

Tested and verified

- OpenShift Java Runtime
- JBoss Web Server (Tomcat) Embedded Container
- JPA, JAX-RS, Health Checks, SSO and more

Kubernetes를 이용한 배포(istio in the future)

- Spring-cloud-kubernetes project

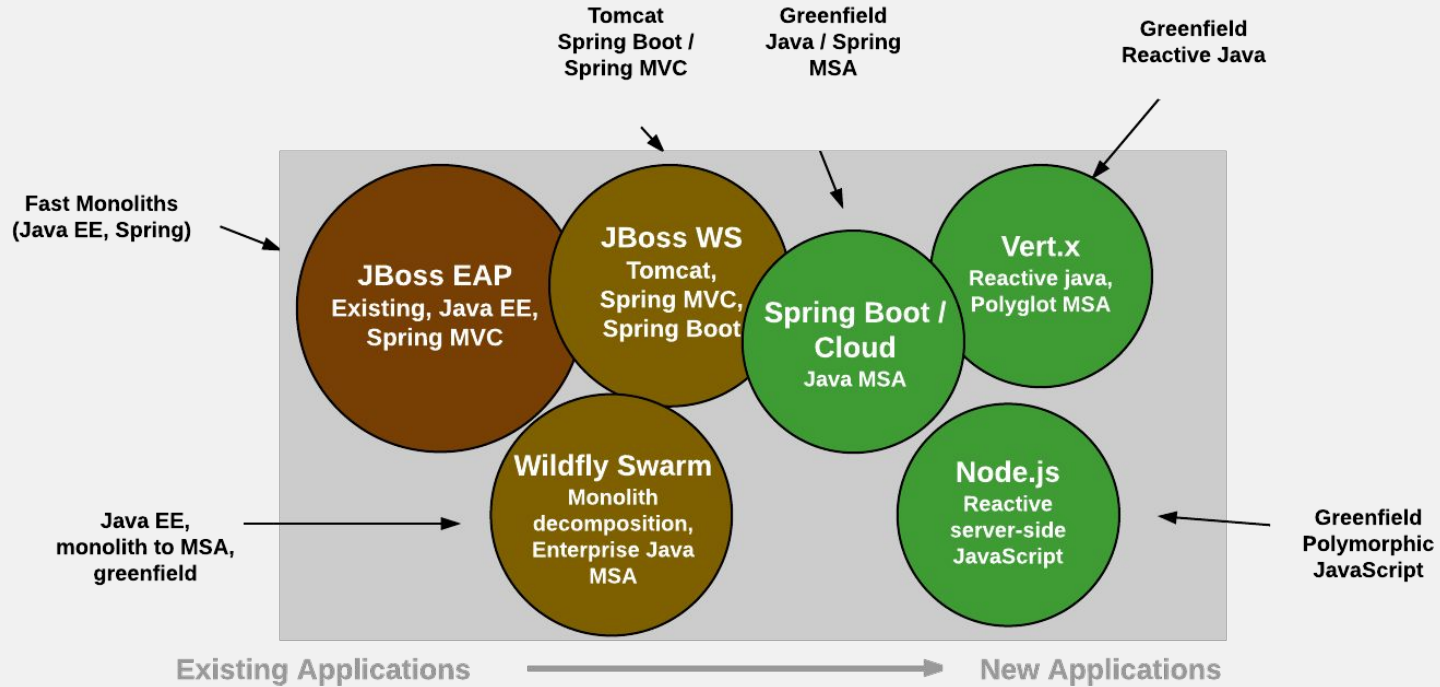
Provide a “supported” stack

- Pivotal 서비스를 대체한 Red Hat support 제공

RHOAR RUNTIMES USE CASES

Runtimes	Development Target
JBoss EAP	High transaction Java 어플리케이션 및 서비스의 빌드, 배포, 실행을 위한 Open-source Java EE 기반 Application Server
WildFly Swarm	Java 라이브러리 및 부트 스트랩 코드만으로 실행이 가능한 Java Application을 빌드, MicroProfile과의 호환성을 통해 기존 Java EE 환경을 활용하는 Monolithic Application의 Microservice로의 전환 가속
Vert.x	비동기, 비차단 (non-blocking) 개발 모델을 사용하여 분산 및 reactive application을 JVM 위에 구축 Vert.x는 빠른 속도와 낮은 대기 시간을 필요로 하는 Application을 빌드하는 데 적합
Node.js	Reactive, Event-driven 및 non-blocking 서버 JavaScript Application 개발에 Node.js 사용 가능
Tomcat	Spring Boot Application을 실행하기 위한 Tomcat container인 JBoss Web Server 지원

RHOAR RUNTIMES USE CASES - Positioning



RHOAR Launchpad

The screenshot shows the 'Mission' step in the RHOAR Launchpad workflow. A progress bar at the top indicates the current step. The main content area is titled 'Mission' and contains introductory text about mission preconfiguration. Below this, there are three sections, each with a 'Mission proficiency level: Foundational' label:

- CRUD**: Describes the Relational Database Backend Booster as a REST API Level 0 Booster for CRUD operations on a PostgreSQL database.
- Circuit Breaker**: Describes the Circuit Breaker Mission as a generic pattern for reporting and handling service failures.
- Externalized Configuration**: Describes the ConfigMap Mission as a basic example of using ConfigMap for external configuration sources.

Select the Mission

The screenshot shows a green checkmark icon and the text 'Your project is ready!'. Below this, there are five green checkmarks indicating successful steps:

- Creating your new Github repository. Done!
- Pushing your customized Booster code into the repo. Done! We've created your Booster, a starter application, in your new Github repo. To learn about your app and the Mission it fulfills, select it from the available missions documentation.
- Creating your project on OpenShift Online. Done!
- Setting up your build pipeline. Done! Pipelines run your build and deployment in stages; for more information, see the CI section of the docs.
- Configuring to trigger builds on Git pushes. Done! Webhooks let OpenShift Online know that you've pushed changes to your Github repository, and will automatically trigger a new pipeline build to deploy them.

Deploy on Openshift

The screenshot shows the 'Runtime' step in the RHOAR Launchpad workflow. A progress bar at the top indicates the current step. The main content area is titled 'Runtime' and contains introductory text about runtime framework choices. Below this, there are three cards representing different runtime options:

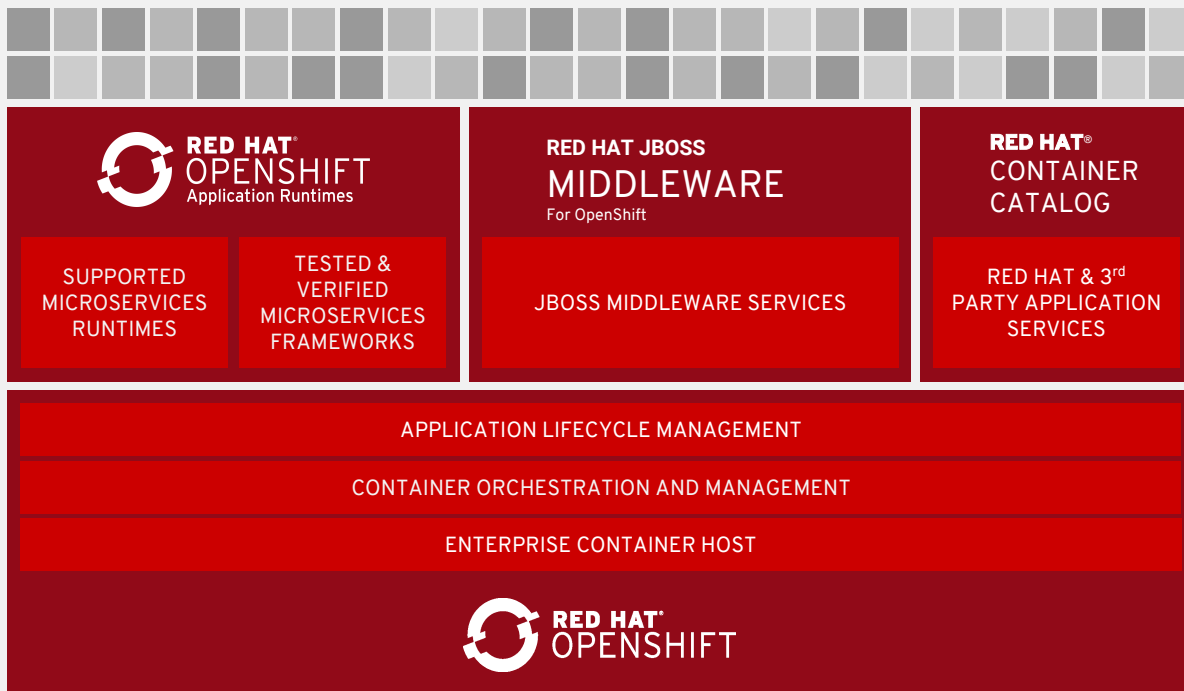
- VERTX**: Eclipse Vert.x
- Spring Boot**
- WildFly Swarm**

Choose a Runtime

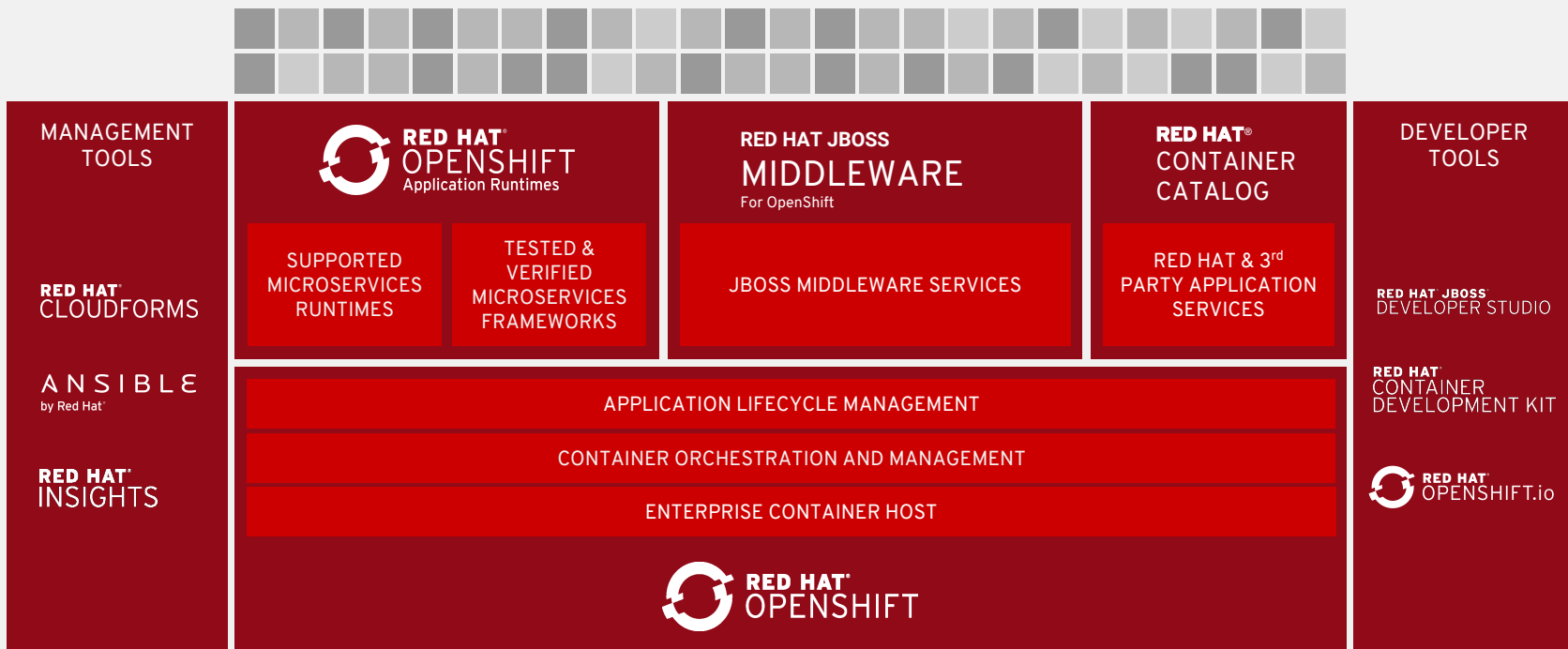
The screenshot shows the OpenShift console interface for an application named 'anonymous-vertxcrud'. The left sidebar contains navigation options: Overview, Applications, Builds, Resources, Storage, and Monitoring. The main content area displays details for the 'CRUD VERTX' application, including deployment configurations and service status. It shows 'No deployments' for the 'crud-vertx' service and 'No grouped services' for both 'crud-vertx' and 'my-database'.

Run & Use

THE PATH TO CLOUD NATIVE DEVELOPMENT WITH RED HAT



THE PATH TO CLOUD NATIVE DEVELOPMENT WITH RED HAT





THANK YOU

