Integration architecture with Java EE and Spring

Markus Eisele (Lightbend)
Josh Long (Pivotal)

@myfear

Lightbend

Josh Long (龙之春)

the Spring Developer Advocate

Introduction

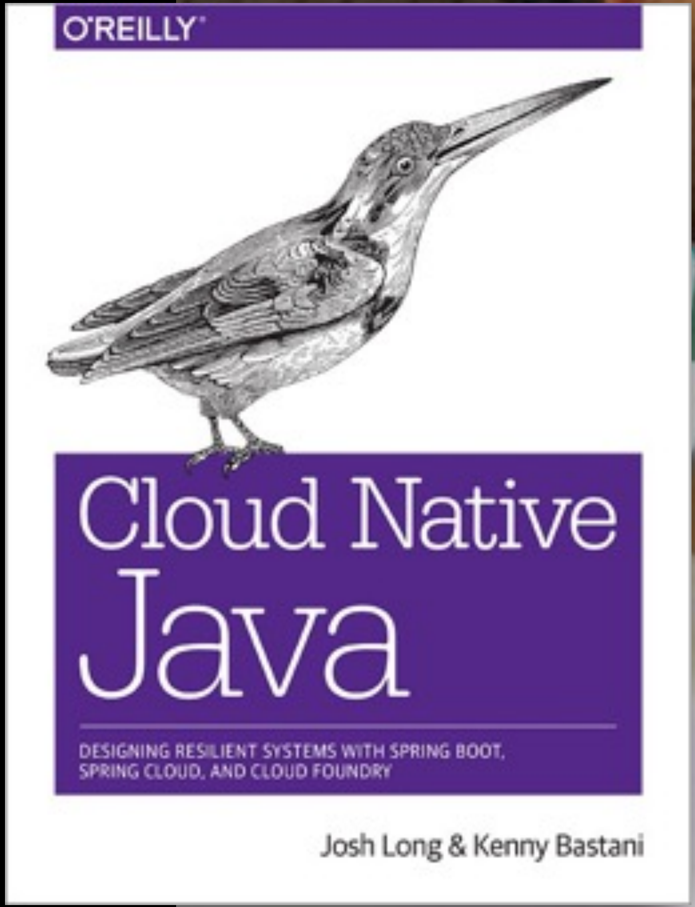
Building Microservices with Spring Boot LiveLessons

with Josh Long & Phillip Webb

Addison-Wesley

livelessons

video instruction from technology experts



O'REILLY

Cloud Native Java

DESIGNING RESILIENT SYSTEMS WITH SPRING BOOT, SPRING CLOUD, AND CLOUD FOUNDRY

Josh Long & Kenny Bastani

- http://**cloudnativejava.io**
- **@starbuxman**
- **josh@joshlong.com**
- **Java Champion**
- **open-source contributor**
  (Spring Boot, Spring Cloud, Spring Integration, Vaadin, Activiti, etc etc)

Hands-on

Get your environment ready

http://**bit.ly/1MpEaS5**

Where We’ve Been

Motivations for Java EE / J2EE

- Centralized Infrastructures
- Shared baseline
- Centralized governance and management
- Innovation through implementation
- Convention over configuration

Today’s reality for Java EE

- Shared baseline installs no longer relevant
- Customized and distributed fat-jars.
- Innovation can’t be standardized
- Centralized governance vs. DevOps
- Interesting for commodity (e.g. JDBC)

Motivations for Spring

- Spring was born to simplify J2EE APIs
- Spring was born to promote testing, faster feedback loops
- to provide patterns and best practices
- provide flexibility through configuration (over convention)

Rod Johnson

Rod Johnson is an enterprise Java architect specializing in scalable web applications. He has worked with both Java and J2EE since their release, and he is a member of JSR 154 Expert Group defining the Servlet 2.4 specification.

Updates, source code, and Wrox technical support at www.wrox.com

Today’s reality for Spring

- Java EE (vs J2EE) is *very* concise, powerful
- continuous improvement and delivery still key to the power of Spring
- Best practices evolve, and Spring has tried to keep up. Meanwhile, old *best practices..* aren’t
- Spring has tried to learn from Java EE and Rails by accommodating smart conventions.

And now?

Java EE

>

<

==

?

Spring

Why this talk?

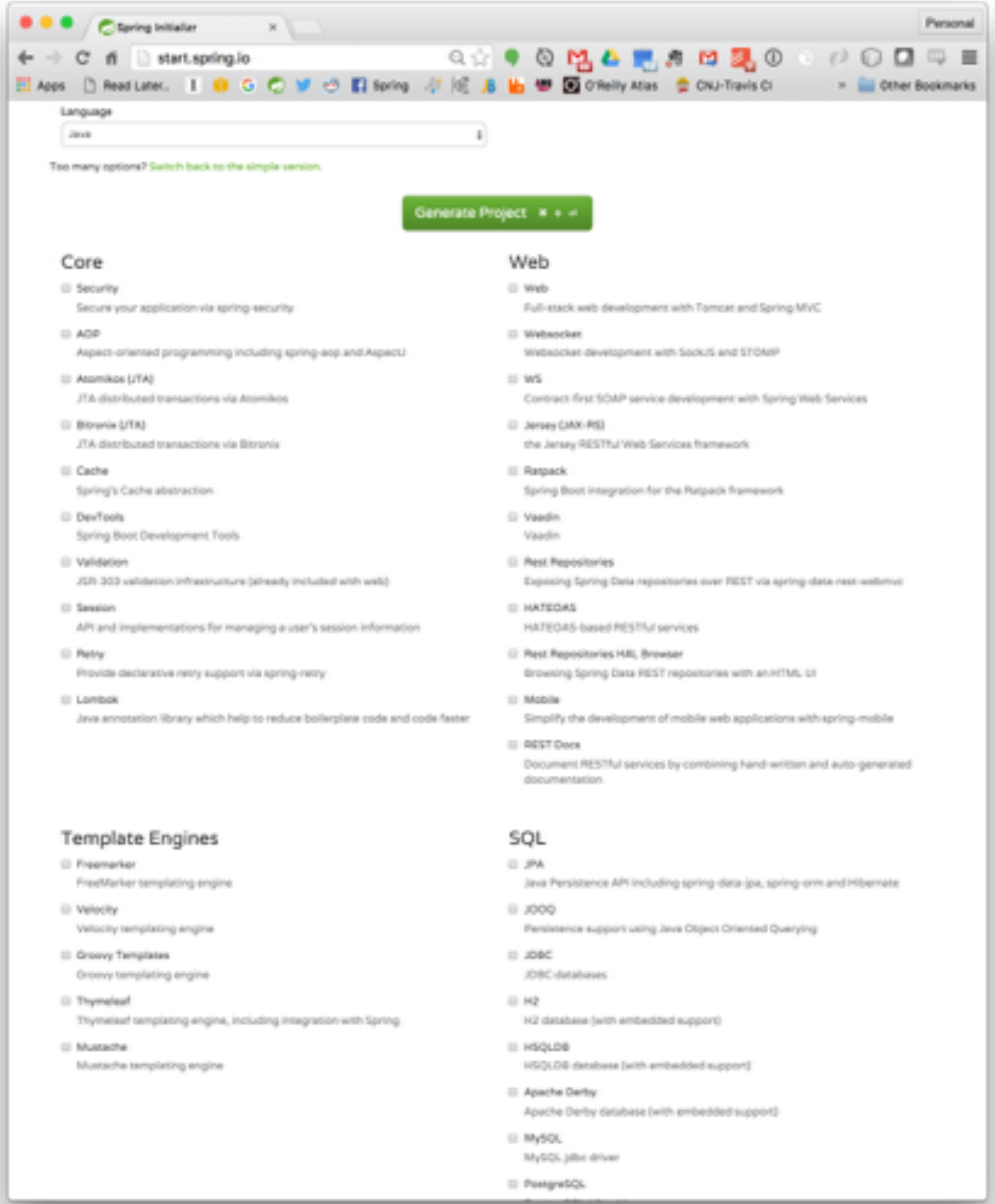
Use Spring APIs from **Java EE**

Motivation

- Features that Java EE doesn’t provide out-of-the-box
  - Spring Security
  - Social login
  - the `JdbcTemplate`
  - MVC
  - NoSQL
  - Enterprise Application Integration
  - big data
  - RabbitMQ
  - NoSQL
  - Kafka
  - …. maybe it’s better to show!

Motivation

start.spring.io



Language

Java

Too many options? Switch back to the simple version.

Core

- ☐ Security
  Secure your application via spring-security
- ☐ AOP
  Aspect-oriented programming including spring-aop and AspectJ
- ☐ Atomikos (JTA)
  JTA distributed transactions via Atomikos
- ☐ Bitronix (JTA)
  JTA distributed transactions via Bitronix
- ☐ Cache
  Spring's Cache abstraction
- ☐ DevTools
  Spring Boot Development Tools
- ☐ Validation
  JSR-303 validation infrastructure (already included with web)
- ☐ Session
  API and implementations for managing a user's session information
- ☐ Retry
  Provide declarative retry support via spring-retry
- ☐ Lombok
  Java annotation library which help to reduce boilerplate code and code faster

Web

- ☐ Web
  Full-stack web development with Tomcat and Spring MVC
- ☐ Websocket
  Websocket development with SockJS and STOMP
- ☐ WS
  Contract-first SOAP service development with Spring Web Services
- ☐ Jersey (JAX-RS)
  the Jersey RESTful Web Services framework
- ☐ Ratpack
  Spring Boot integration for the Ratpack framework
- ☐ Vaadin
  Vaadin
- ☐ Rest Repositories
  Exposing Spring Data repositories over REST via spring-data-rest-webmvc
- ☐ HATEOAS
  HATEOAS-based RESTful services
- ☐ Rest Repositories HAL Browser
  Browsing Spring Data REST repositories with an HTML UI
- ☐ Mobile
  Simplify the development of mobile web applications with spring-mobile
- ☐ REST Docs
  Document RESTful services by combining hand-written and auto-generated documentation

Template Engines

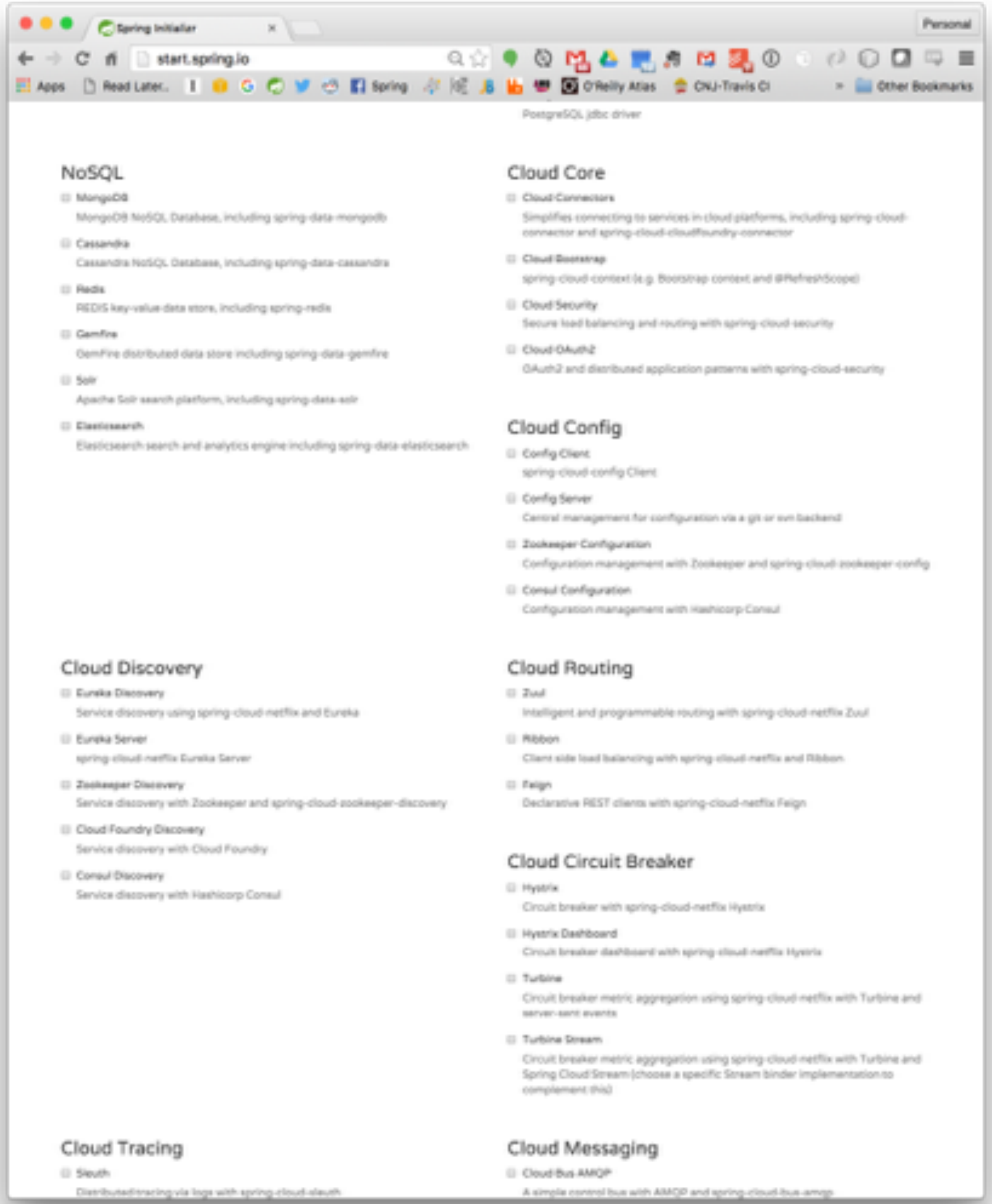
- ☐ Freemarker
  FreeMarker templating engine
- ☐ Velocity
  Velocity templating engine
- ☐ Groovy Templates
  Groovy templating engine
- ☐ Thymeleaf
  Thymeleaf templating engine, including integration with Spring
- ☐ Mustache
  Mustache templating engine

SQL

- ☐ JPA
  Java Persistence API including spring-data-jpa, spring-orm and Hibernate
- ☐ JOOQ
  Persistence support using Java Object Oriented Querying
- ☐ JDBC
  JDBC databases
- ☐ H2
  H2 database (with embedded support)
- ☐ HSQLDB
  HSQLDB database (with embedded support)
- ☐ Apache Derby
  Apache Derby database (with embedded support)
- ☐ MySQL
  MySQL jdbc driver
- ☐ PostgreSQL

Motivation

start.spring.io



PostgreSQL jdbc driver

NoSQL

☐ MongoDB
MongoDB NoSQL Database, including spring-data-mongodb

☐ Cassandra
Cassandra NoSQL Database, including spring-data-cassandra

☐ Redis
REDIS key-value data store, including spring-redis

☐ Gemfire
GemFire distributed data store including spring-data-gemfire

☐ Solr
Apache Solr search platform, including spring-data-solr

☐ Elasticsearch
Elasticsearch search and analytics engine including spring-data-elasticsearch

Cloud Core

☐ Cloud Connectors
Simplifies connecting to services in cloud platforms, including spring-cloud-connector and spring-cloud-cloudfoundry-connector

☐ Cloud Bootstrap
spring-cloud-context (e.g. Bootstrap context and @RefreshScope)

☐ Cloud Security
Secure load balancing and routing with spring-cloud-security

☐ Cloud OAuth2
OAuth2 and distributed application patterns with spring-cloud-security

Cloud Config

☐ Config Client
spring-cloud-config Client

☐ Config Server
Central management for configuration via a git or svn backend

☐ Zookeeper Configuration
Configuration management with Zookeeper and spring-cloud-zookeeper-config

☐ Consul Configuration
Configuration management with Hashicorp Consul

Cloud Discovery

☐ Eureka Discovery
Service discovery using spring-cloud-netflix and Eureka

☐ Eureka Server
spring-cloud-netflix Eureka Server

☐ Zookeeper Discovery
Service discovery with Zookeeper and spring-cloud-zookeeper-discovery

☐ Cloud Foundry Discovery
Service discovery with Cloud Foundry

☐ Consul Discovery
Service discovery with Hashicorp Consul

Cloud Routing

☐ Zuul
Intelligent and programmable routing with spring-cloud-netflix Zuul

☐ Ribbon
Client side load balancing with spring-cloud-netflix and Ribbon

☐ Feign
Declarative REST clients with spring-cloud-netflix Feign

Cloud Circuit Breaker

☐ Hystrix
Circuit breaker with spring-cloud-netflix Hystrix

☐ Hystrix Dashboard
Circuit breaker dashboard with spring-cloud-netflix Hystrix

☐ Turbine
Circuit breaker metric aggregation using spring-cloud-netflix with Turbine and server-sent events

☐ Turbine Stream
Circuit breaker metric aggregation using spring-cloud-netflix with Turbine and Spring Cloud Stream (choose a specific Stream binder implementation to complement this)

Cloud Tracing

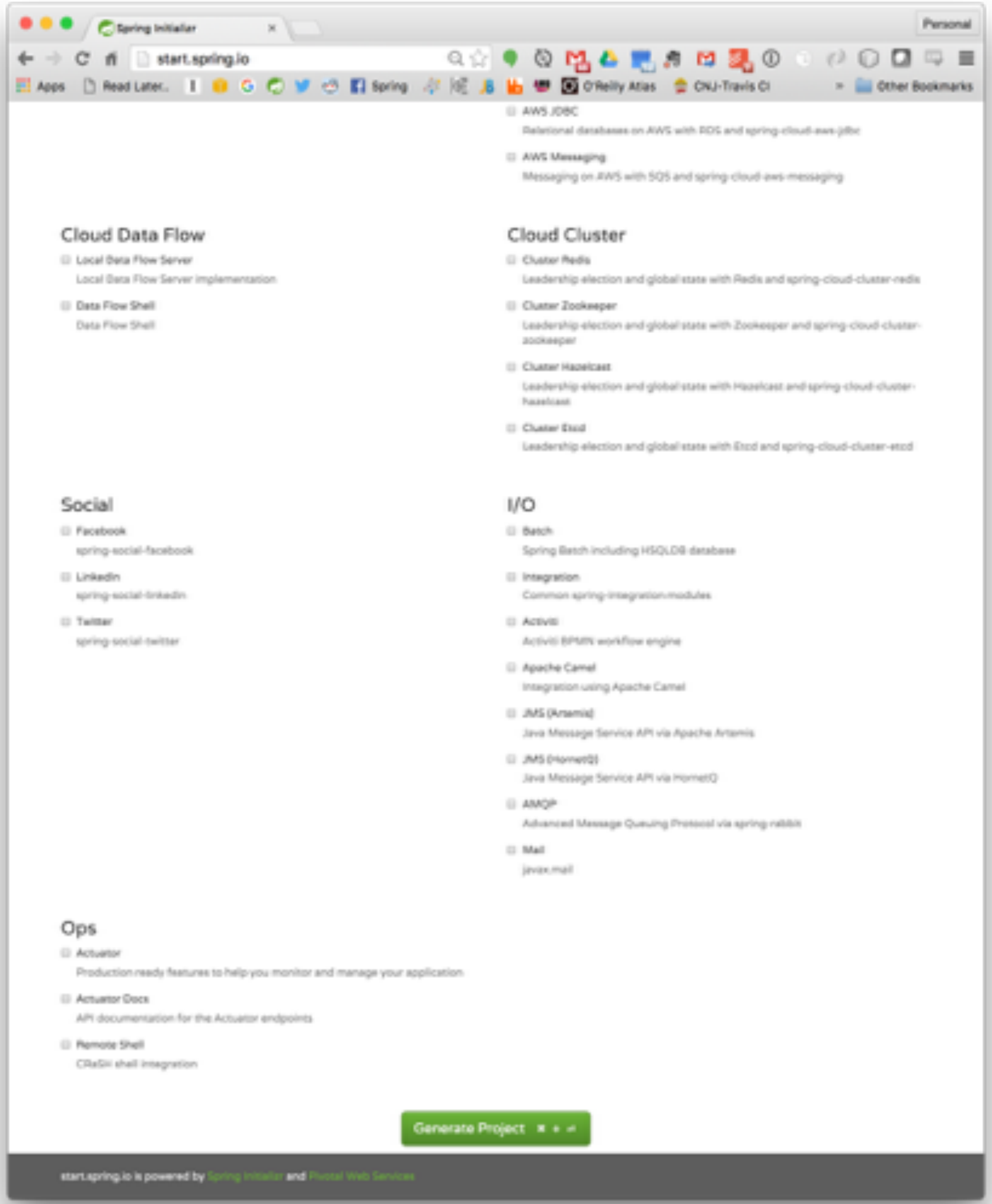
☐ Sleuth
Distributed tracing via logs with spring-cloud-sleuth

Cloud Messaging

☐ Cloud Bus AMQP
A simple control bus with AMQP and spring-cloud-bus-amqp

Motivation

start.spring.io



AWS JDBC
Relational databases on AWS with RDS and spring-cloud-aws-jdbc

AWS Messaging
Messaging on AWS with SQS and spring-cloud-aws-messaging

Cloud Data Flow

Local Data Flow Server
Local Data Flow Server implementation

Data Flow Shell
Data Flow Shell

Cloud Cluster

Cluster Redis
Leadership election and global state with Redis and spring-cloud-cluster-redis

Cluster Zookeeper
Leadership election and global state with Zookeeper and spring-cloud-cluster-zookeeper

Cluster Hazelcast
Leadership election and global state with Hazelcast and spring-cloud-cluster-hazelcast

Cluster Etcd
Leadership election and global state with Etcd and spring-cloud-cluster-etcd

Social

Facebook
spring-social-facebook

LinkedIn
spring-social-linkedin

Twitter
spring-social-twitter

I/O

Batch
Spring Batch including HSQLDB database

Integration
Common spring-integration modules

Activiti
Activiti BPMN workflow engine

Apache Camel
Integration using Apache Camel

JMS (Artemis)
Java Message Service API via Apache Artemis

JMS (HornetQ)
Java Message Service API via HornetQ

AMQP
Advanced Message Queuing Protocol via spring-rabbit

Mail
javax.mail

Ops

Actuator
Production ready features to help you monitor and manage your application

Actuator Docs
API documentation for the Actuator endpoints

Remote Shell
CRaSH shell integration

start.spring.io is powered by Spring Initializr and Pivotal Web Services

Motivation

- Backwards Compatibility
  - Spring framework has a very long tail: Spring framework 4 runs on Servlet 2.5+ (2006!!), Java EE 6 (2009) and Java 6+.
  - Websphere 7 and WebLogic 10.3.4 require JPA 2 feature packs

demo

Coffee Break

Use Java EE APIs from **Spring**

Motivation

- Want to or have to migrate
- Your team already has the knowledge
- You want to use standards where standards make sense because they’re commoditized or invasive:
  - JTA, JPA, JSR303, JSR 330, JCA, JDBC, JMS, Servlets, etc. etc.

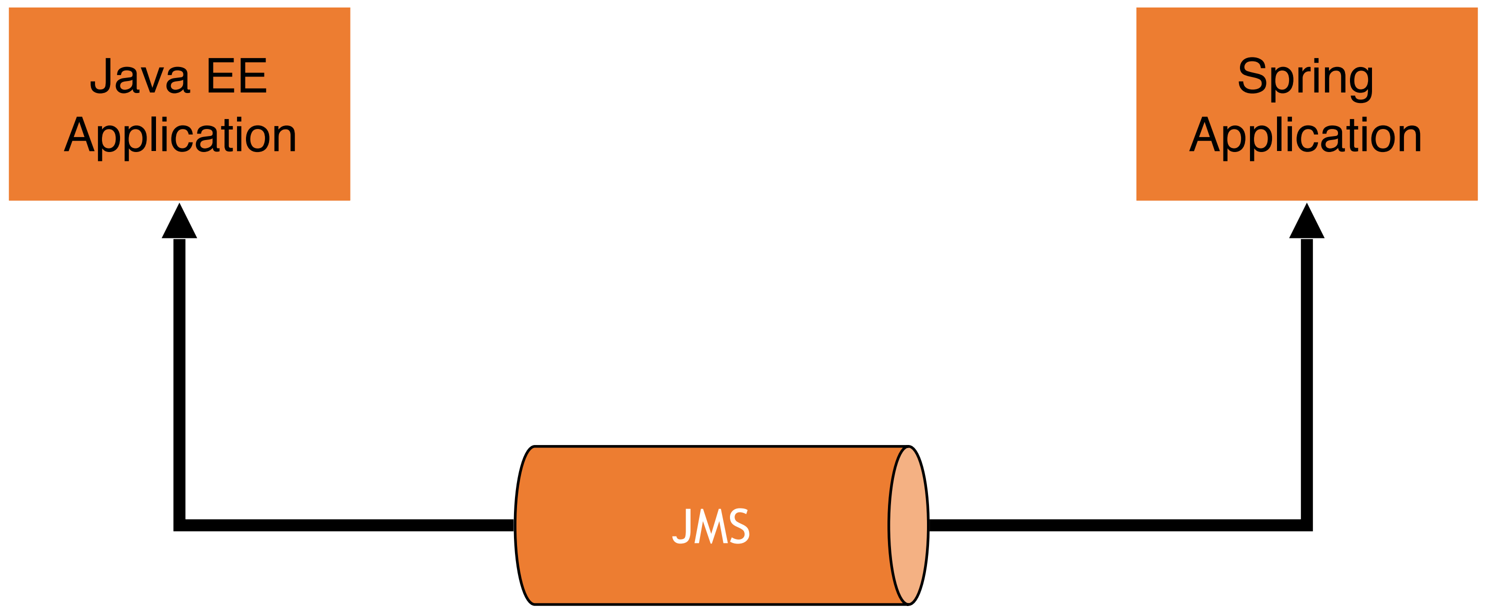
demo

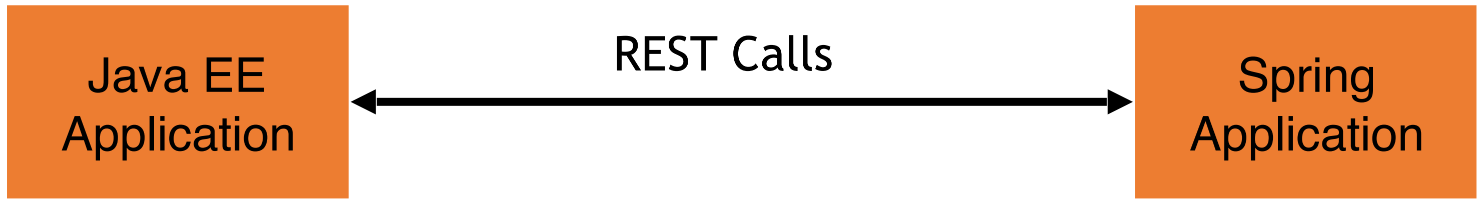
Option **3**

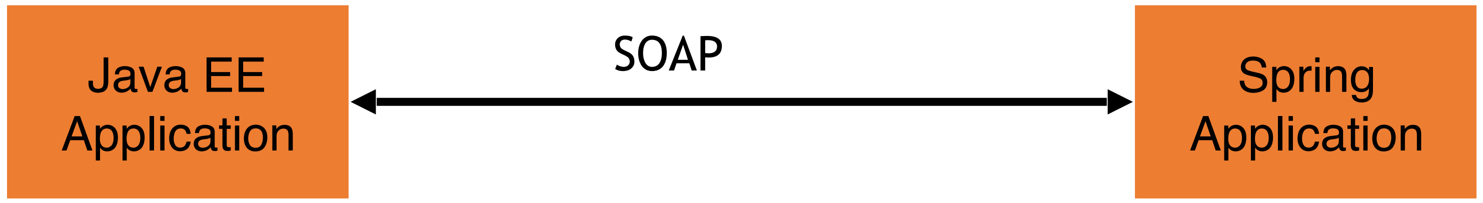
Integrate both worlds

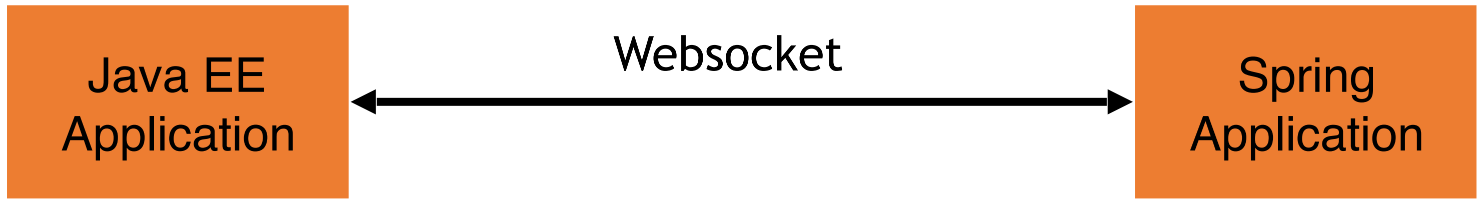
Motivation

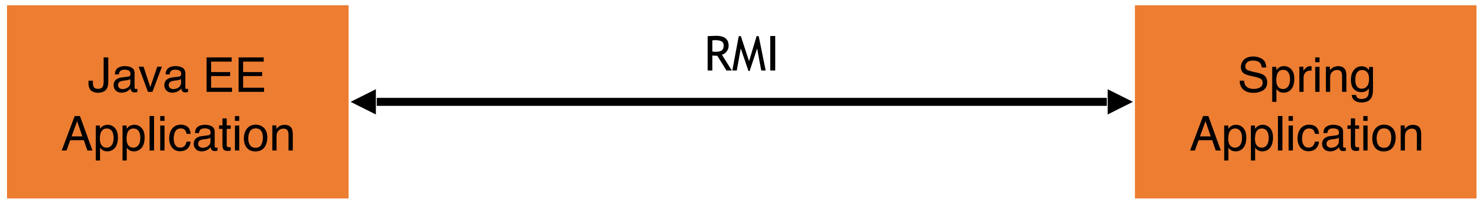
- Today’s world is growing more and more polyglot and heterogeneous.
- Open Source is driving innovation.
- Closed source stays platform decision.
- Parts move over. Others don’t.

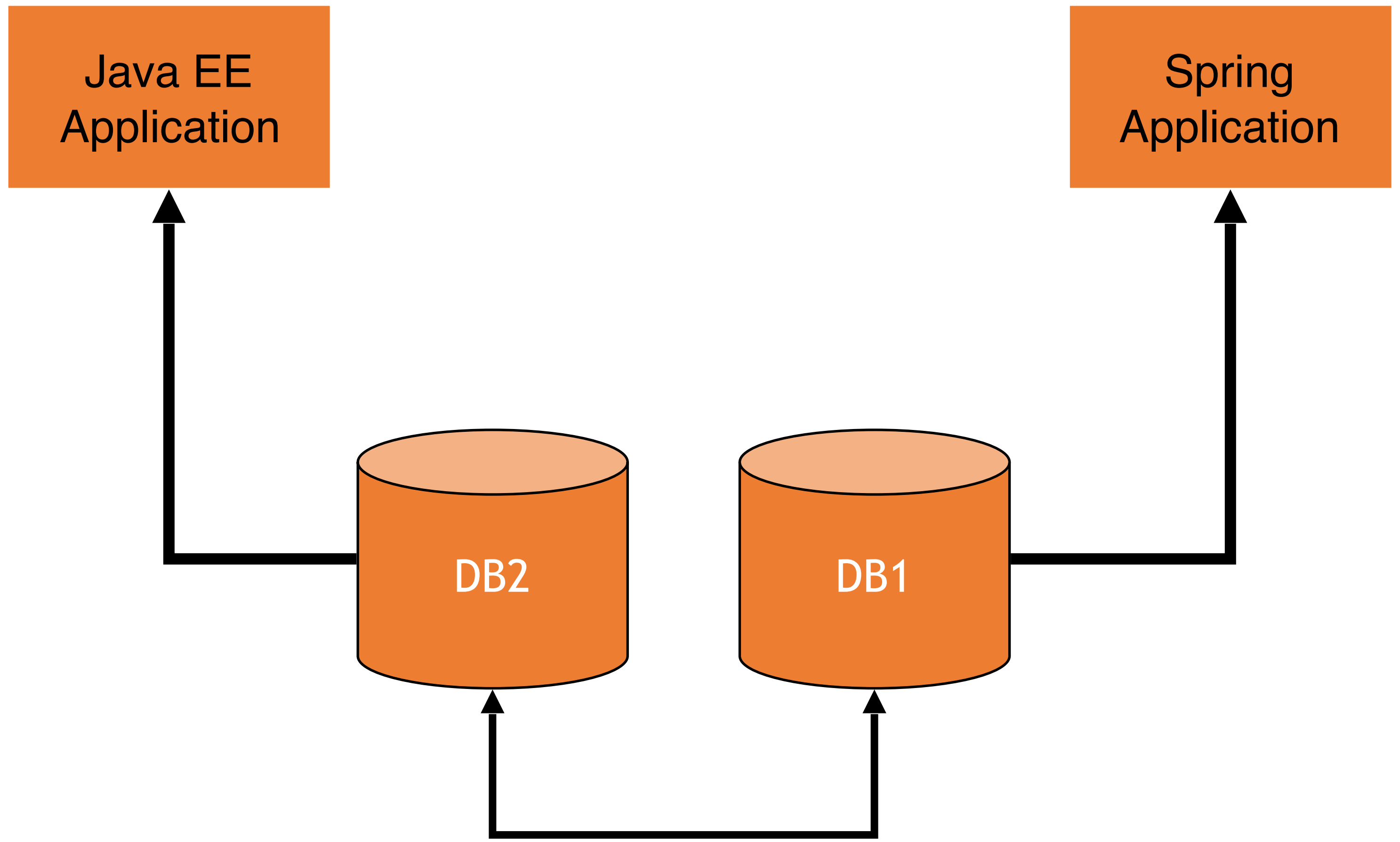

Java EE Application
Spring Application
JMS

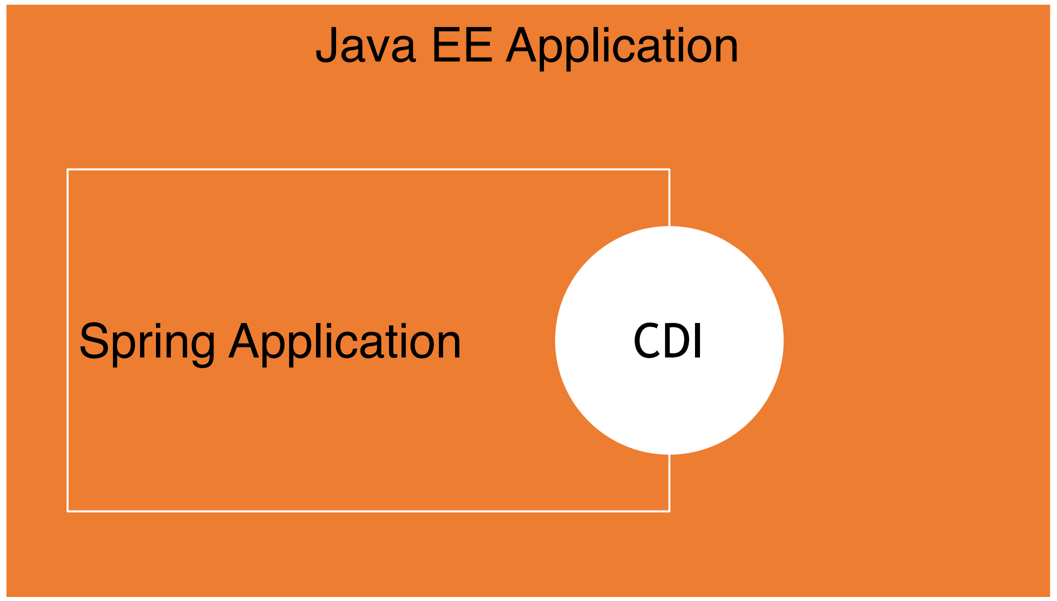

Java EE Application
REST Calls
Spring Application


Java EE Application
SOAP
Spring Application


Java EE Application
Websocket
Spring Application


Java EE Application
RMI
Spring Application


Java EE Application
Spring Application
DB2
DB1


Java EE Application
Spring Application
CDI

micro services

/ˈmʌɪkrəʊ/

noun

Dropwizard

spring io

lagom

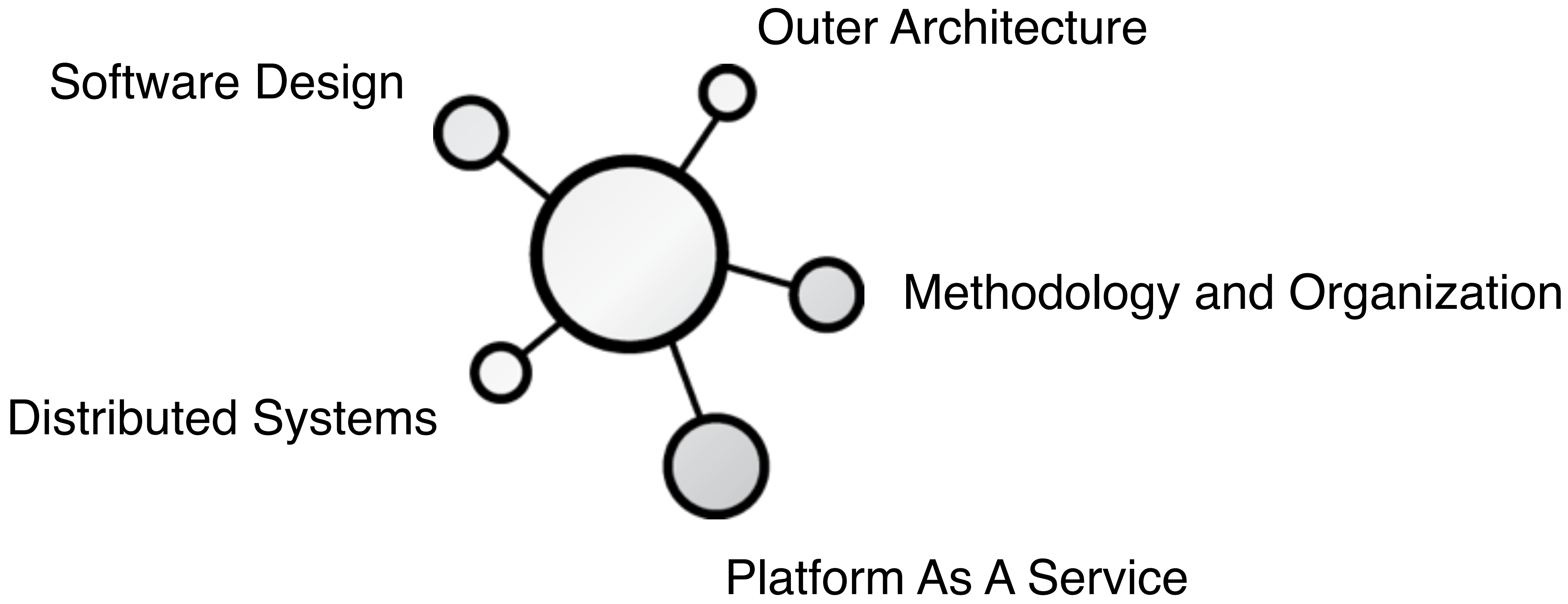
WildFly SWARM

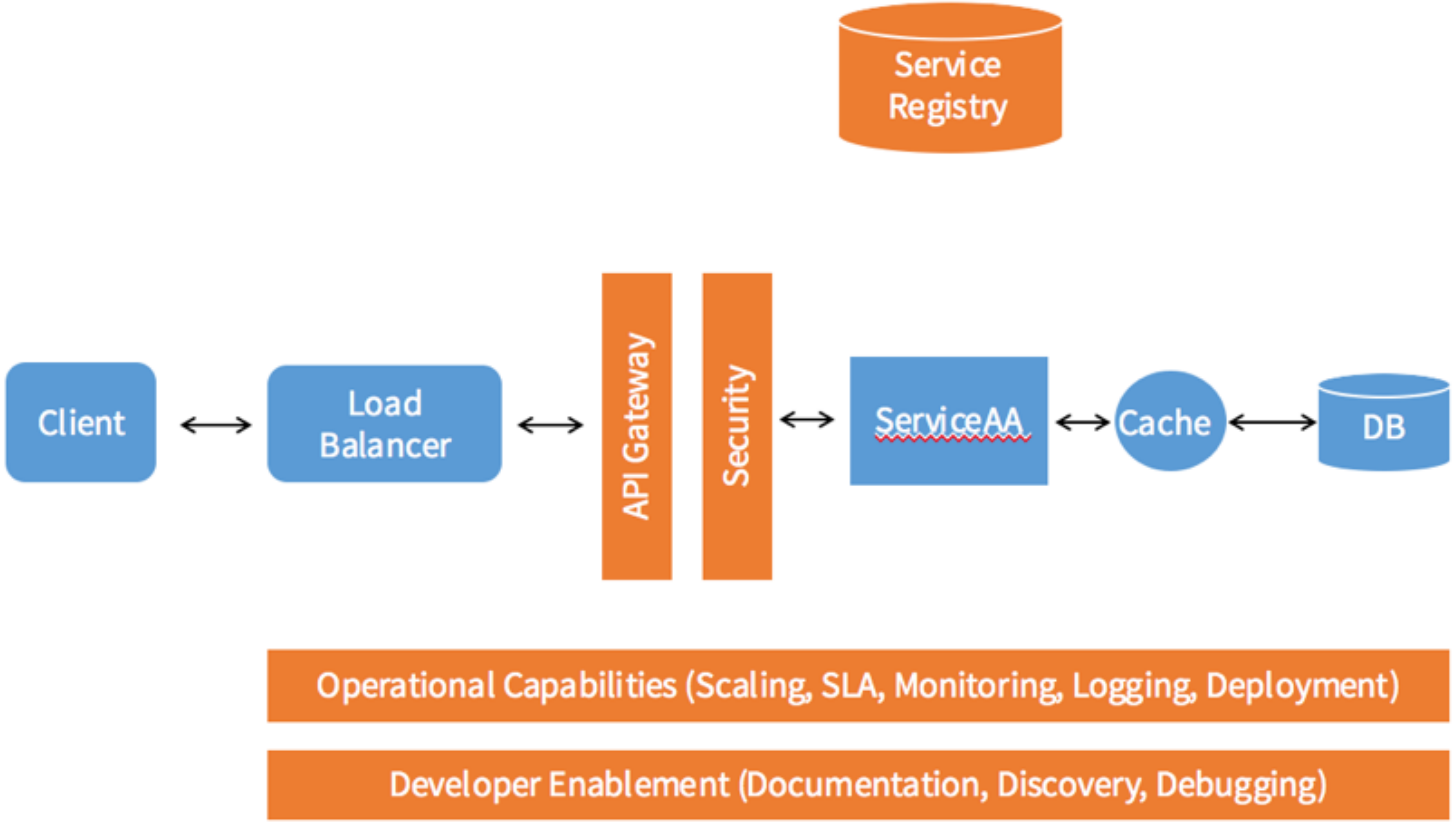
play

Grails

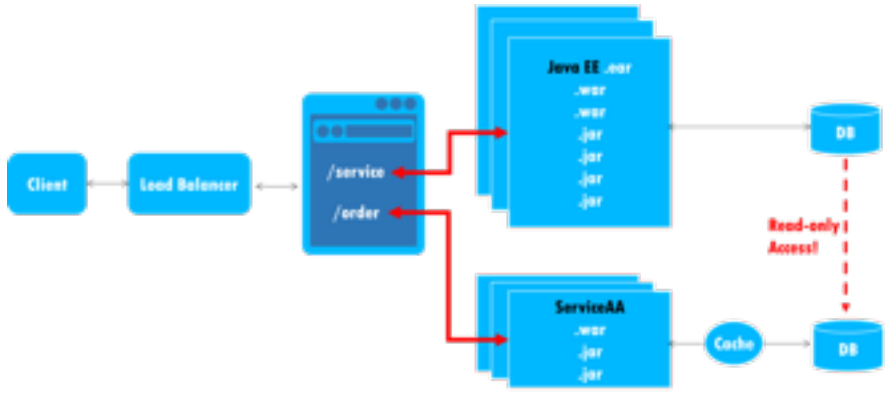
kumuluzEE

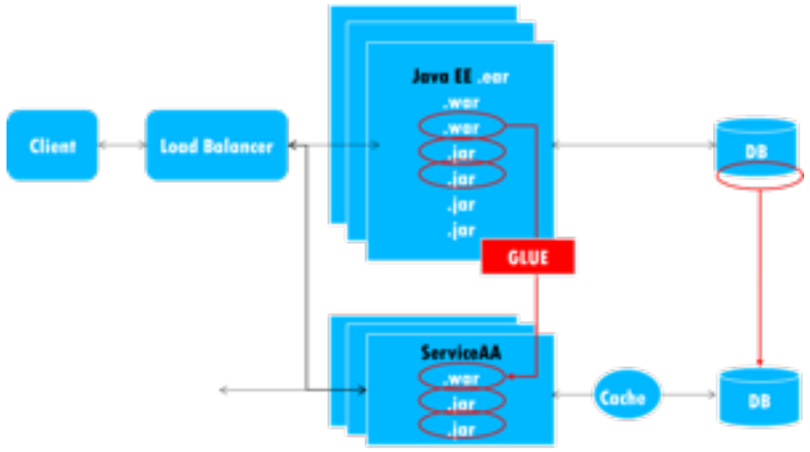
payara

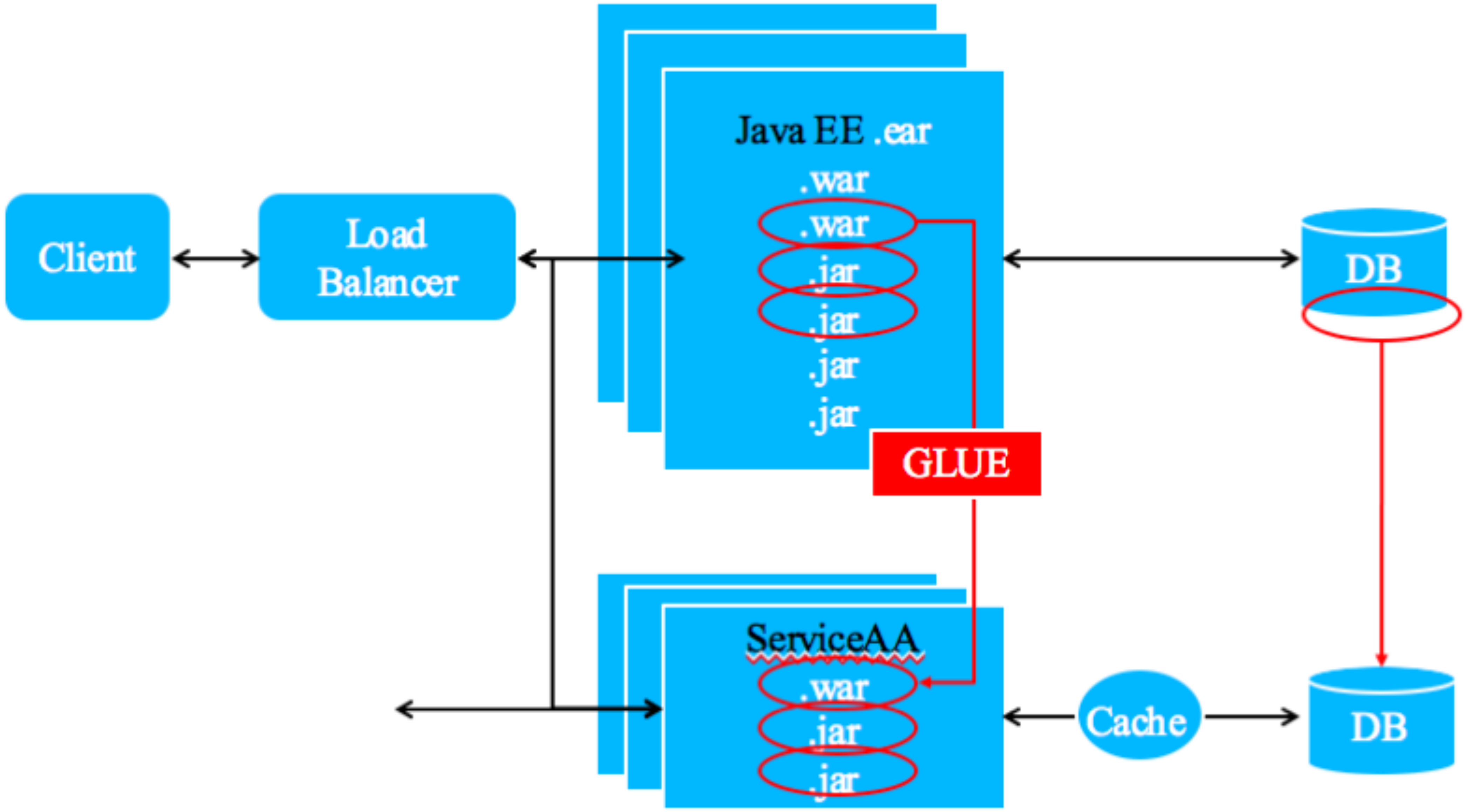

Outer Architecture
Software Design
Methodology and Organization
Distributed Systems
Platform As A Service

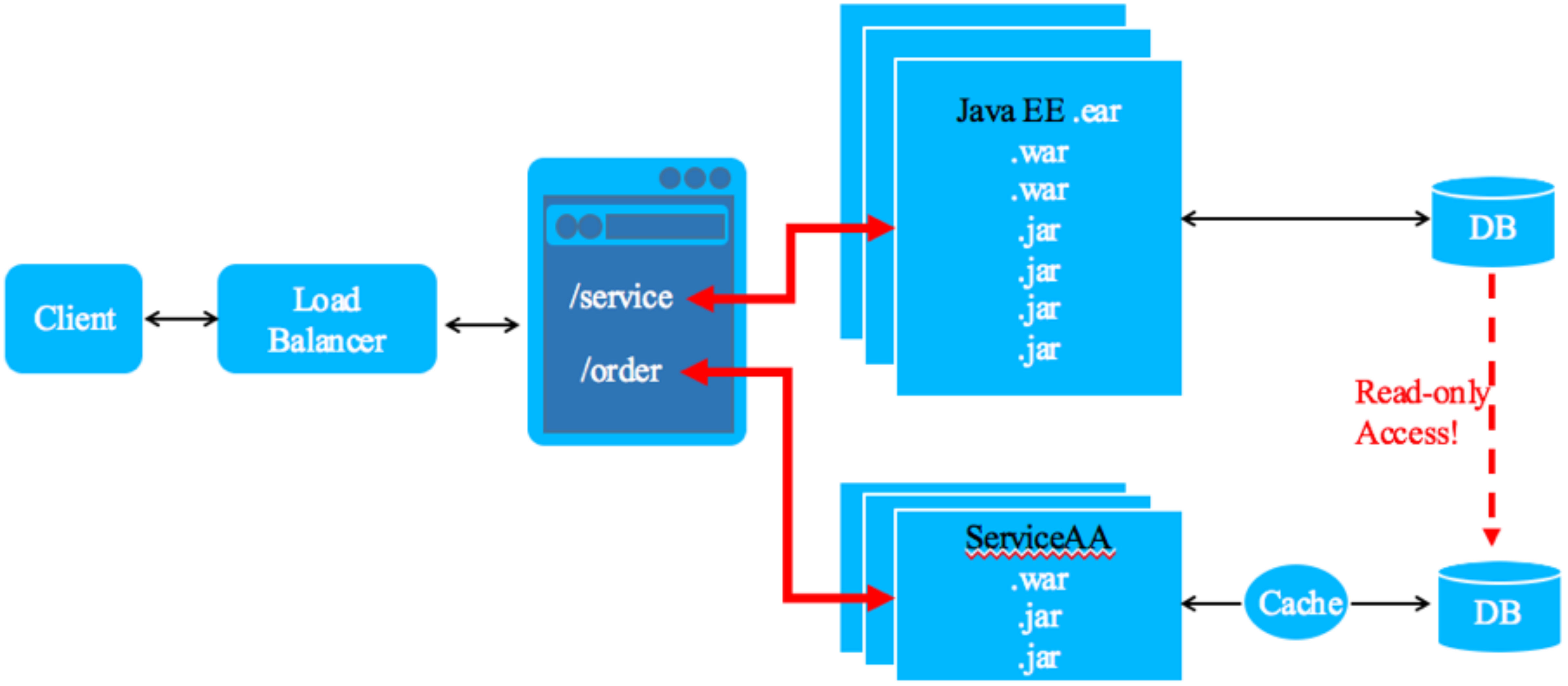

Service Registry
Client
Load Balancer
API Gateway
Security
ServiceAA
Cache
DB
Operational Capabilities (Scaling, SLA, Monitoring, Logging, Deployment)
Developer Enablement (Documentation, Discovery, Debugging)

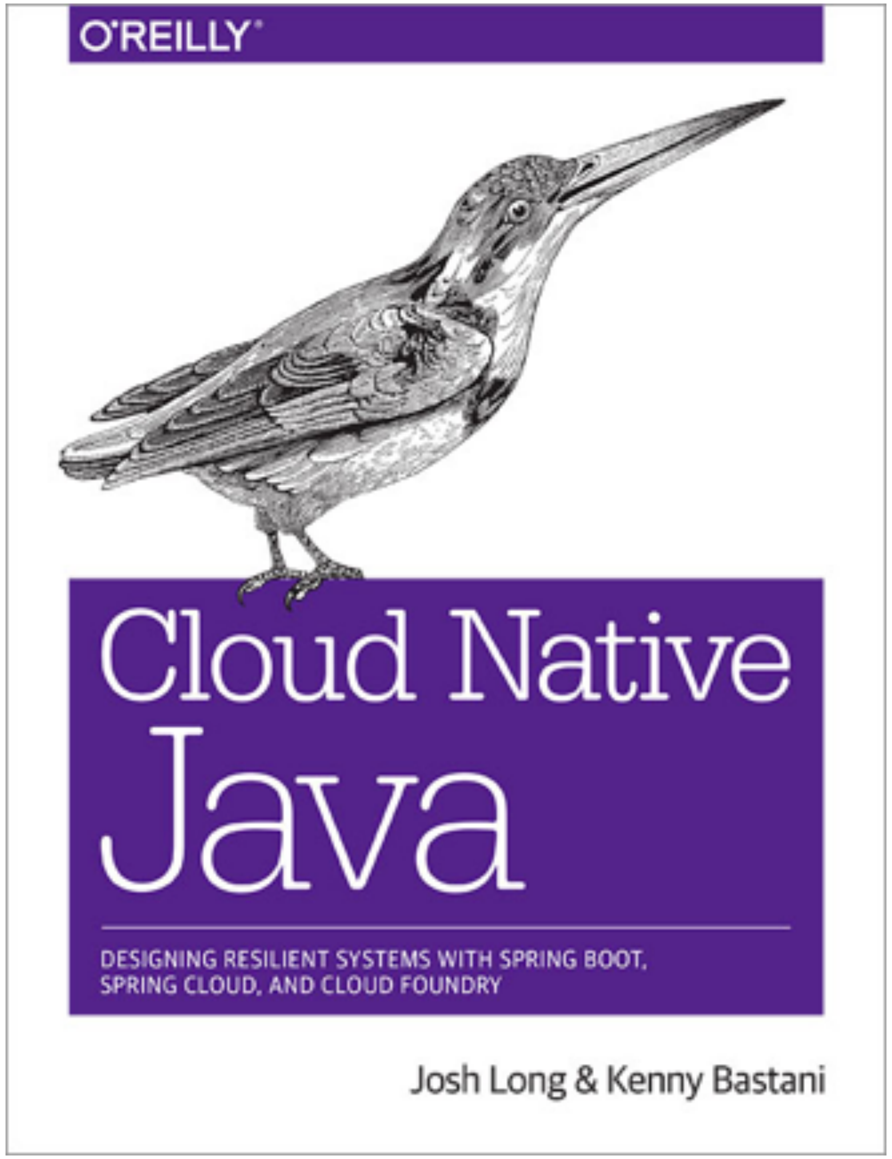
Migration **Approaches**






Client
Load Balancer
Java EE .ear
.war
.war
.jar
.jar
.jar
.jar
GLUE
DB
ServiceAA
.war
.jar
.jar
Cache
DB


Client
Load Balancer
/service
/order
Java EE .ear
.war
.war
.jar
.jar
.jar
.jar
DB
Read-only Access!
ServiceAA
.war
.jar
.jar
Cache
DB


O'REILLY

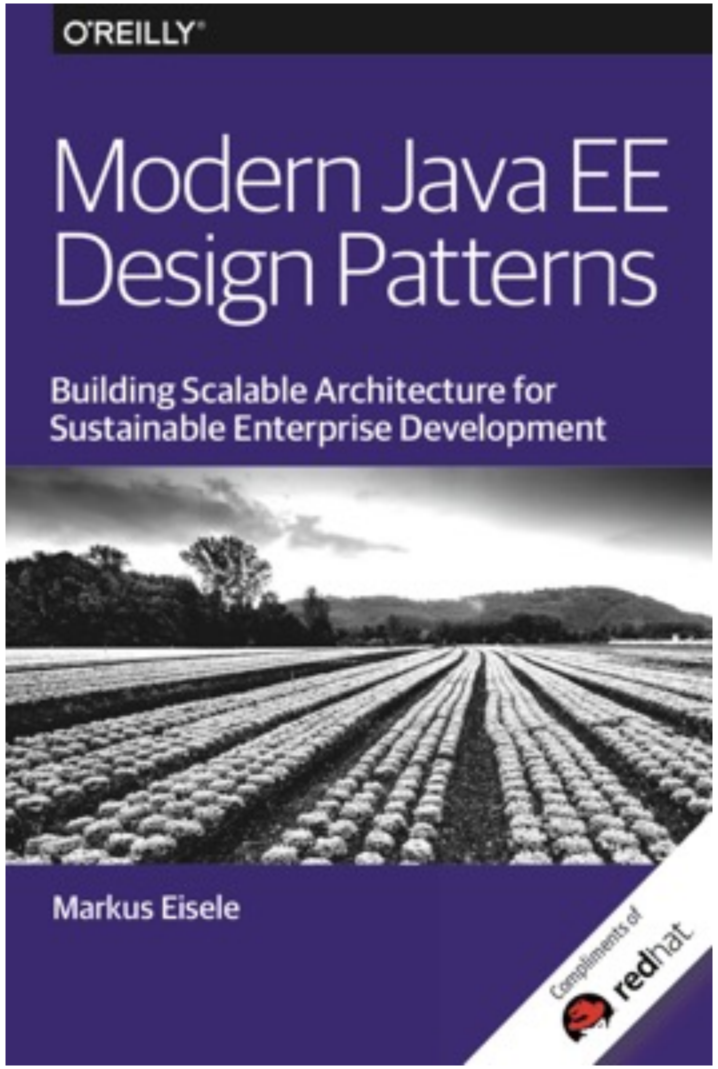
Cloud Native Java

DESIGNING RESILIENT SYSTEMS WITH SPRING BOOT, SPRING CLOUD, AND CLOUD FOUNDRY

Josh Long & Kenny Bastani

- **evolve or die**
- build software that survives, scales and evolves on a dynamic cloud environment

http://cloudnativejava.io/about/


O'REILLY
Modern Java EE Design Patterns
Building Scalable Architecture for Sustainable Enterprise Development
Markus Eisele
Compliments of redhat

- Understand the challenges of starting a greenfield development vs tearing apart an existing brownfield application into services
- Examine your business domain to see if microservices would be a good fit
- Explore best practices for automation, high availability, data separation, and performance
- Align your development teams around business capabilities and responsibilities
- Inspect design patterns such as aggregator, proxy, pipeline, or shared resources to model service interactions

http://bit.ly/ModernJavaEE

Thank you.

@myfear @starbuxman