Java 容器化部署

从应用服务器到云原生

张建锋

永源中间件
shihang@useopen.net

GTLC
GLOBAL
TECH LEADERSHIP
CONFERENCE

全球技术领导力峰会

Geekbang. 极客邦科技 | TGO 鲲鹏會

500+高端科技领导者与你一起探讨
技术、管理与商业那些事儿

2019年6月14–15日 | 上海圣诺亚皇冠假日酒店

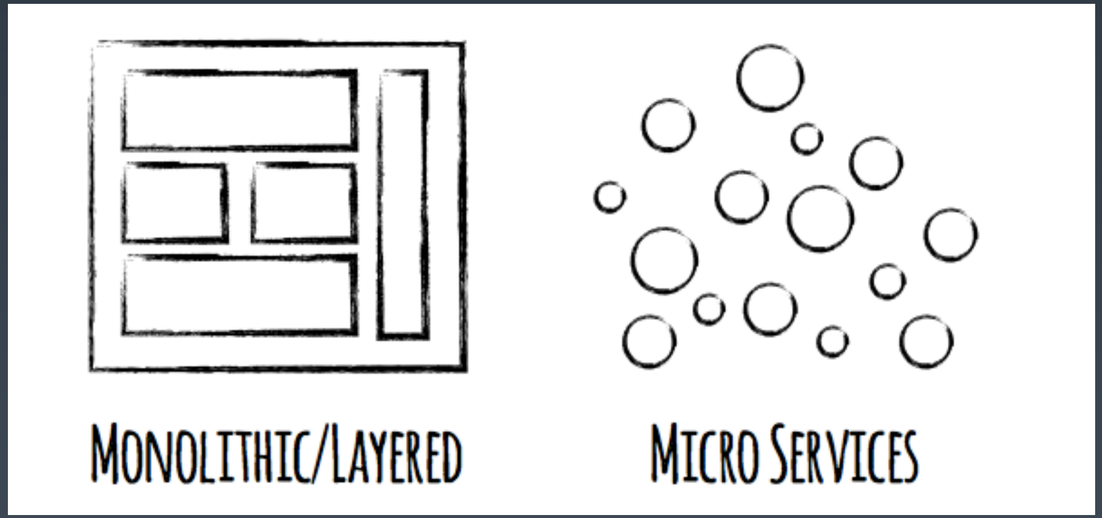
扫码了解更多信息

概述

1. Java 应用程序的部署方式
2. 三个迷思困惑
3. 容器镜像包和软件启动速度
4. Java Cloud Native

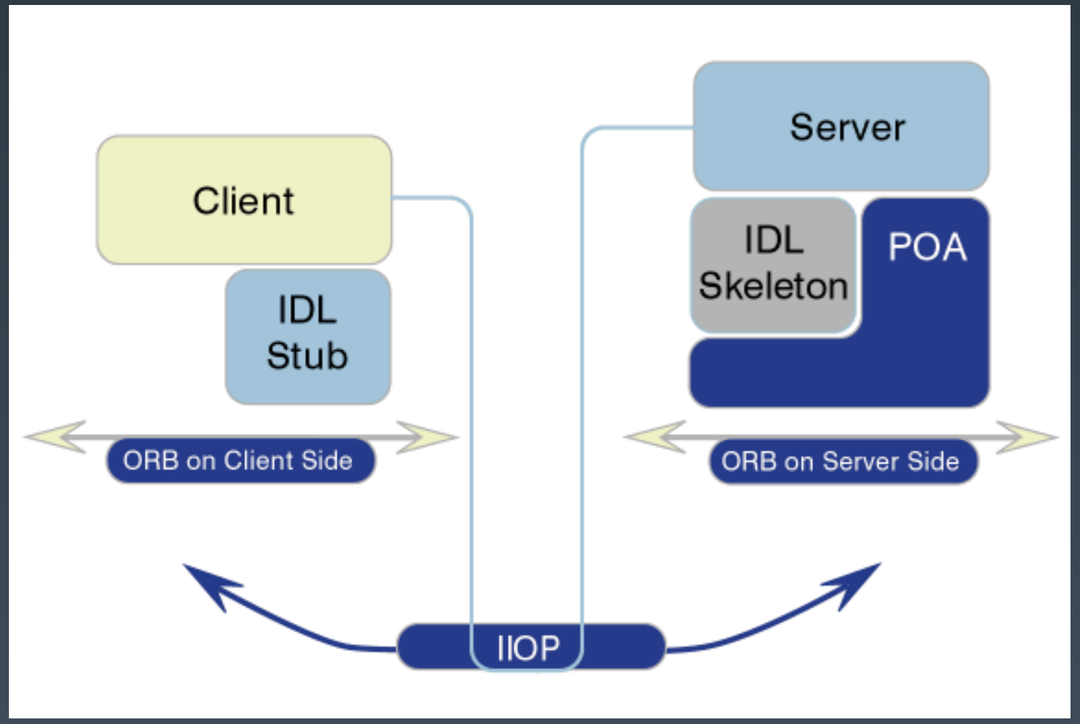
迷思之一： 单体 vs 微服务

1. 哪个在先，哪个在后？
2. 部署包的体积，占用的内存
3. 更容易运维管理
4. 组件间交互和效率



Java刚诞生时

- IDL文件 => 客户端，服务器代码
- 编写服务器实现
- 编译成多个可执行文件
- 多个服务器进程运行
- 注册到命名服务器中
- 集群和高可用
- 编写脚本，维护服务器运行



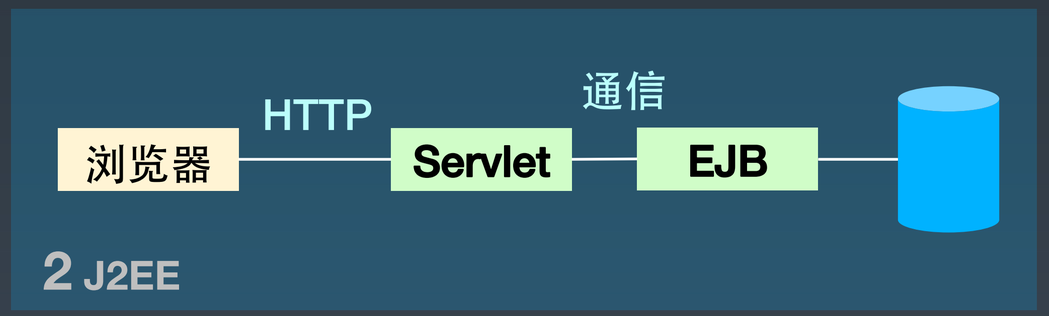
都是微服务架构

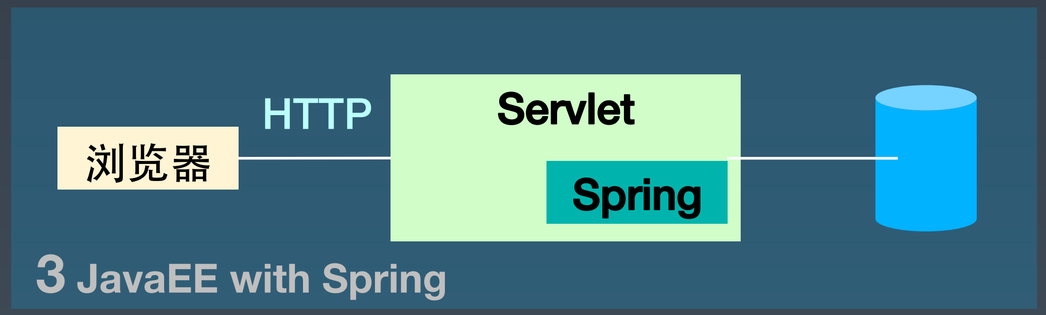
对照表：

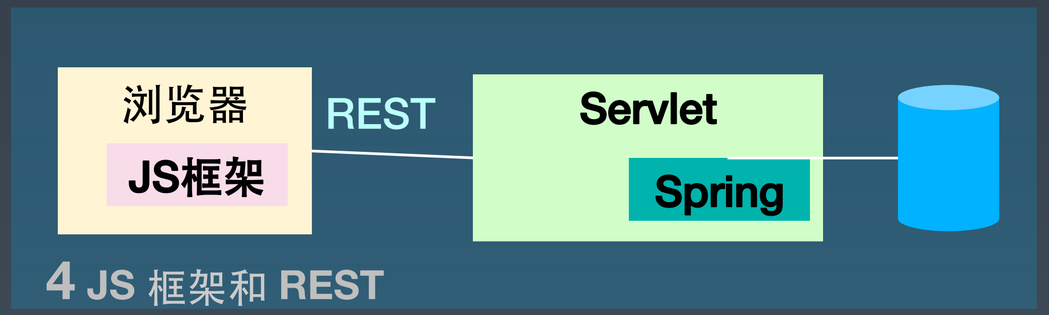
	Corba	gRPC	Dubbo
多种语言	✔	✔	Java
接口定义	IDL	IDL	Java Interface
协议	IIOP	Protobuf/Http2	dubbo等多种
异步响应式	callback	stream	callback
服务治理	–	–	✔

浏览器和前端技术

业界主流技术发展







Spring Framework

基于 Java EE，依赖注入框架，集成了各种技术库

	Java EE	Spring Framework
依赖注入	CDI	IoC Container
AOP	Intercepter	Spring AOP
Persistence	JPA	JPA, JDBC, SpringData
Transaction	JTA, EJB	JTA, JDBC, JPA
Rest	JaxRS	SpringMVC
Messaging	JMS, EJB	Spring Messaging, JMS
Security	JavaEE Security, EJB, Servlet	Spring Security

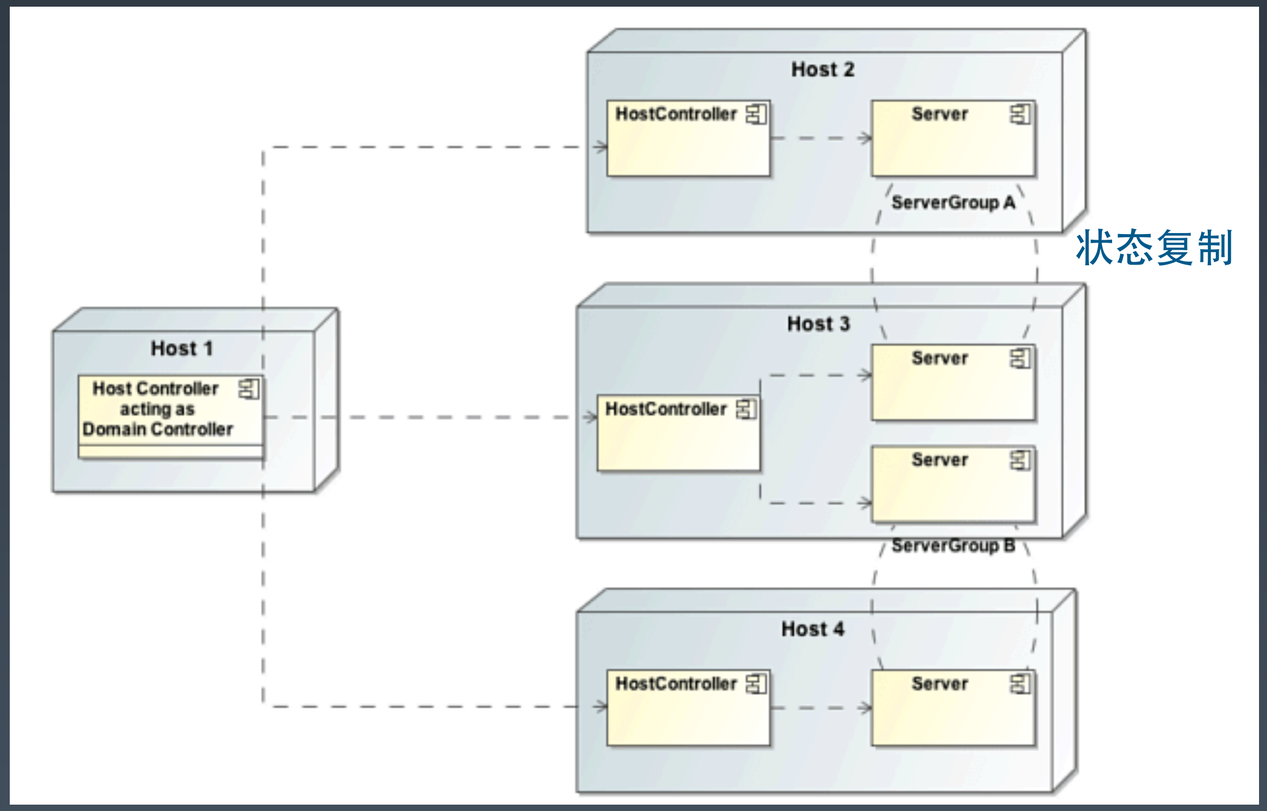
部署包组成

架构

	部署包	是否可执行	应用服务器容器	说明
CS	独立服务器jar	✔	–	客户端维持状态
JEE	ear,war,jar	–	Servlet/EJB	分布对象的争议
Spring web	war	–	Servlet	MVC模式
JS / Rest	war	–	Servlet	异步响应式
Java微服务	war or jar	✔	可选	三个维度划分

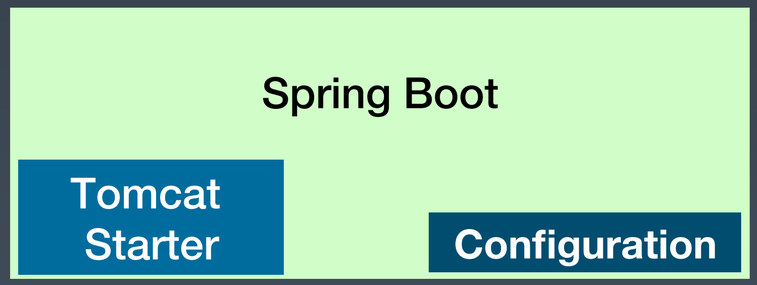
应用服务器集群

- 复制状态
- Servlet 的 Session 信息
- 有状态 Session Bean
- Hibernate 缓存
- Wildfly 利用 Infinispan
- 优化后性能上乘（状态缓存在JVM中）

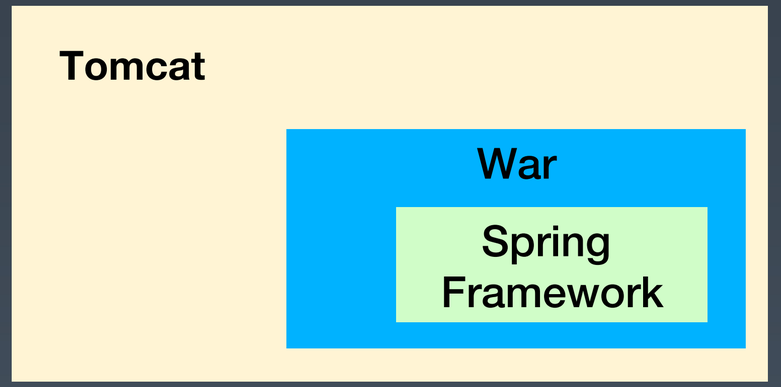


Spring Boot

- 基于 Spring 的可引导的程序
- 可选择不同的 Servlet Starter
- 开发微服务应用
- 与 Servlet 容器包含关系：和Spring框架开发的war包正好相反



VS



微服务要考虑的要素

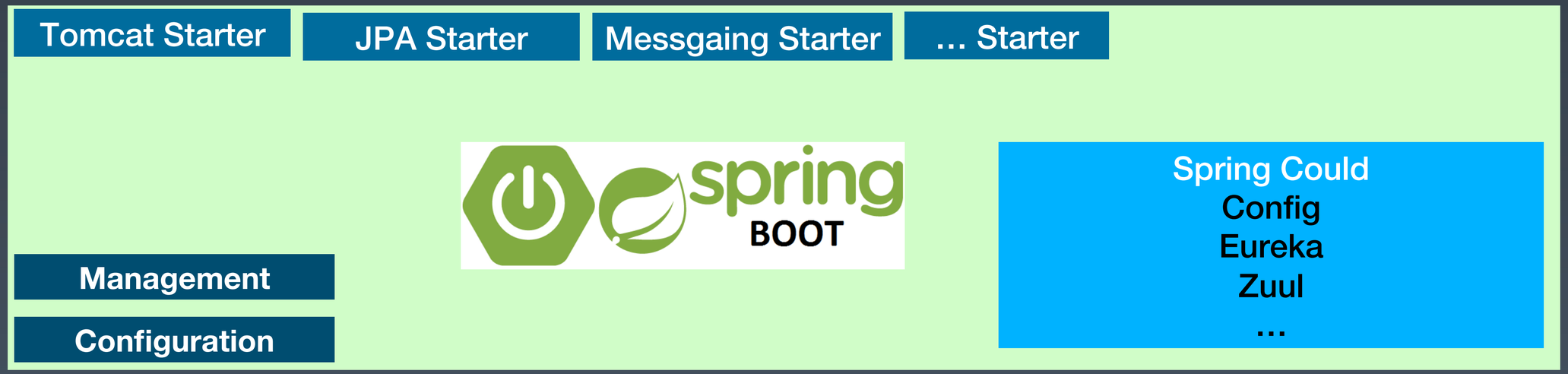
Spring Cloud 和 MicroProfile

微服务要素	Spring Cloud	Microprofile
配置管理	Config server	Config
服务发现	Eureka	JavaEE naming
负载均衡	Ribbon	By Application Server
性能指标	Spring Boot Actuator	Metrics, health
分布式跟踪	Spring Cloud Sleuth	Opentracing
故障容错	Hystrix	Fault-tolerance
路由和拦截器	Zuul	JavaEE Servlet, interceptor

Java微服务

以 Spring 微服务举例：

- JDK + Spring Boot + Spring Cloud + 各种三方库
- Jar 包非常庞大
- 启动时，需要相关服务就绪



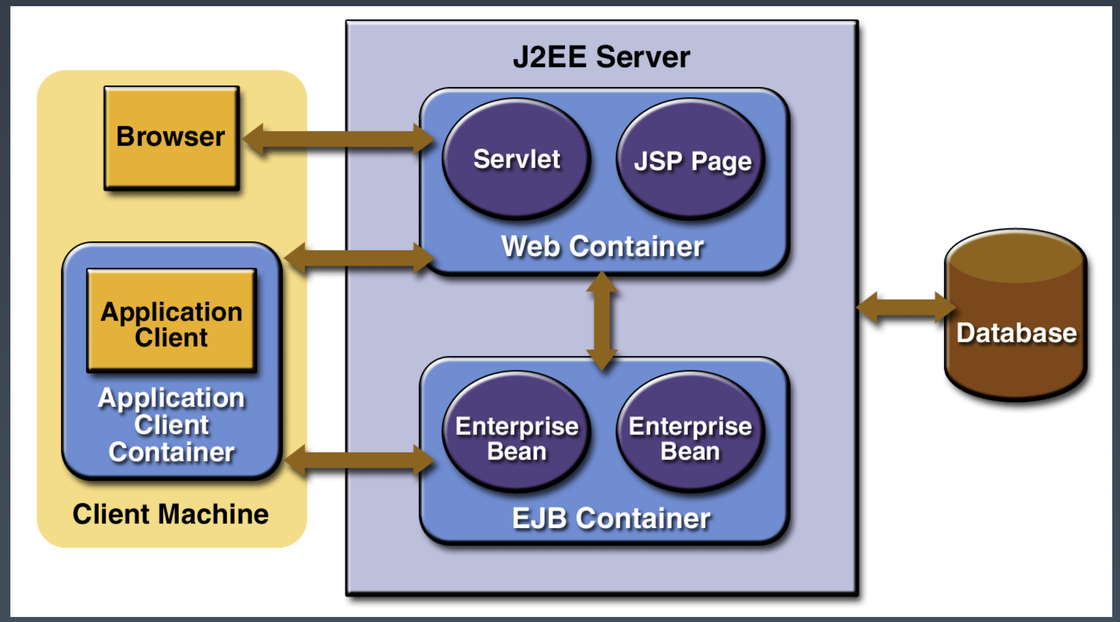
部署包数据对比

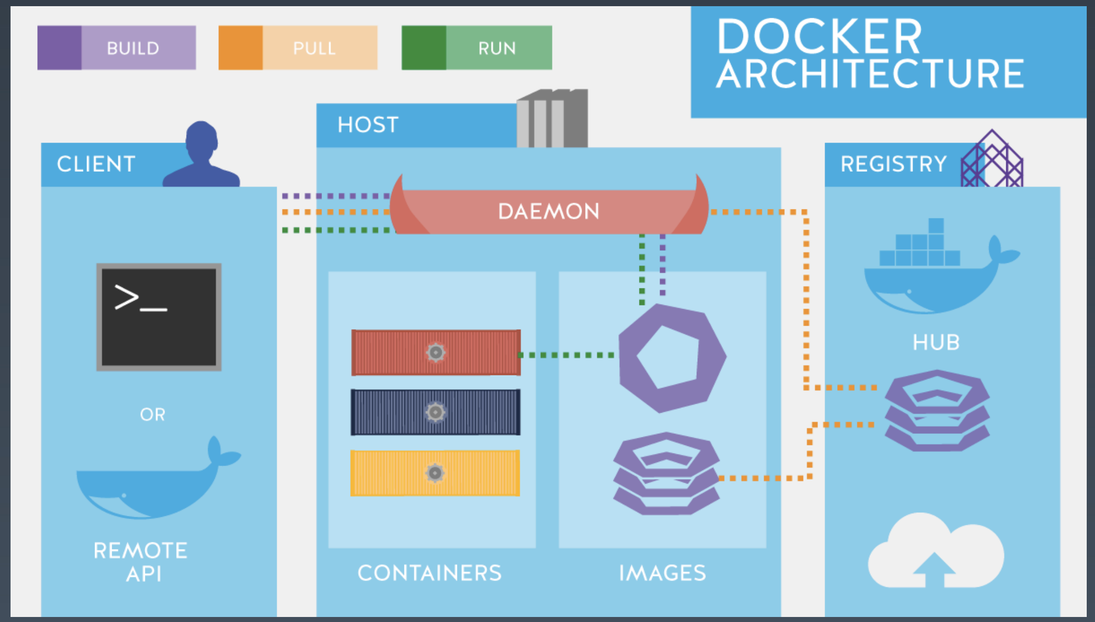
Java 包，功能为返回REST数据信息

方式	JavaEE war包	Spring war包	Spring Boot jar包	Spring Boot / Spring Cloud jar
大小	7K	6.4M	16.7M	39.4M
包含容器总体积	217M	30M	16.7M	39.4M +40.1M(Eureka)
启动时长	4500ms	1800ms	1850ms	7200ms
占用内存	60M	45M	42M	初始>600M，GC后约170M
说明	Wildfly 14 full profile	Tomcat 9, Spring 4.2	SpringBoot 2.1.4, Tomcat 9	SpringCloud 1.4.0, Zuul 1.3

迷思之二： 容器Container所指的是?

Java EE Container 和 Docker container





以进程为单元，资源受限

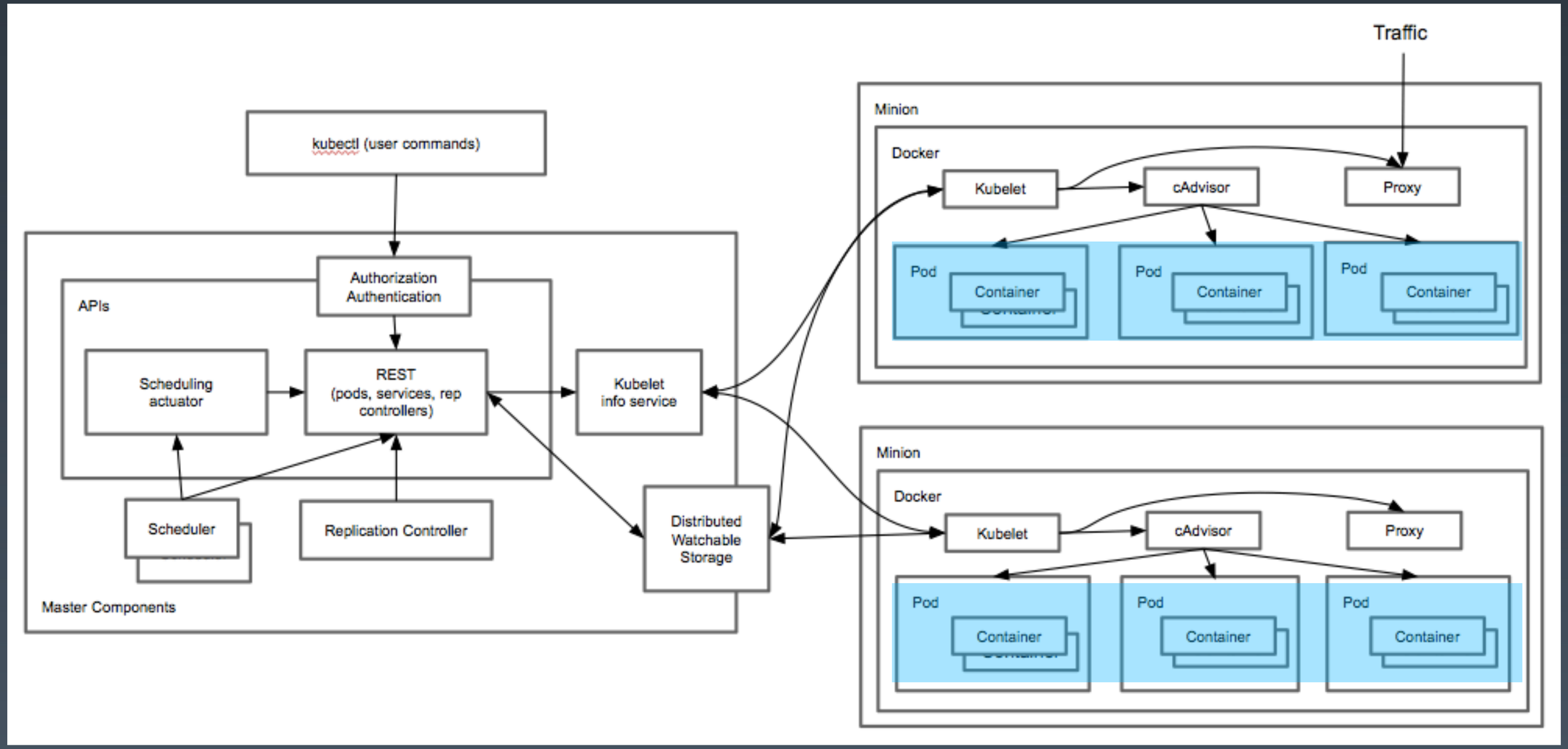
应用容器化部署

应用开发和部署应当：

- 受容器管控
- 向容器设施申请资源
- 提供全局管理功能
- 平台提供接口，方便应用开发

Kubernetes

容器编排，管理大规模集群



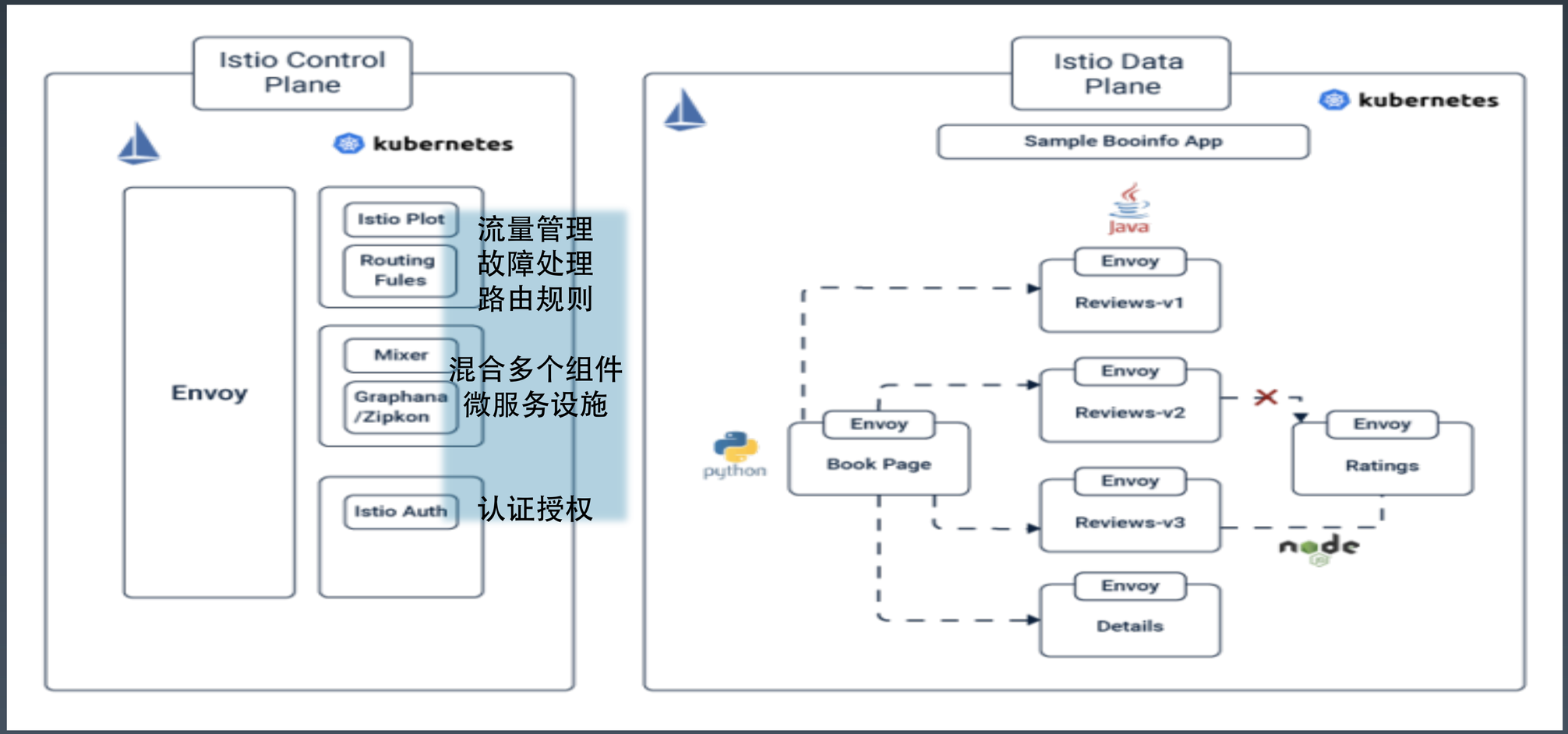
容器部署组件

概念对照

容器	JavaEE应用服务器	Docker and Kubernetes
无状态组件	Managed Bean, Stateless EJB	Pods
有状态组件	Servlet Session, Stateful EJB	StatefulSet
任务	Scheduled Bean	Job, CronJob
主控和任务节点	DomainController, Node	Master, Worker Node
控制器	Controller	Controller manager
服务群组	Server groups	Label selector
扩展性	Extension	CustomResourceDefinitions

Service Mesh帮助微服务“减肥”

提供服务治理功能，业务组件专注逻辑实现

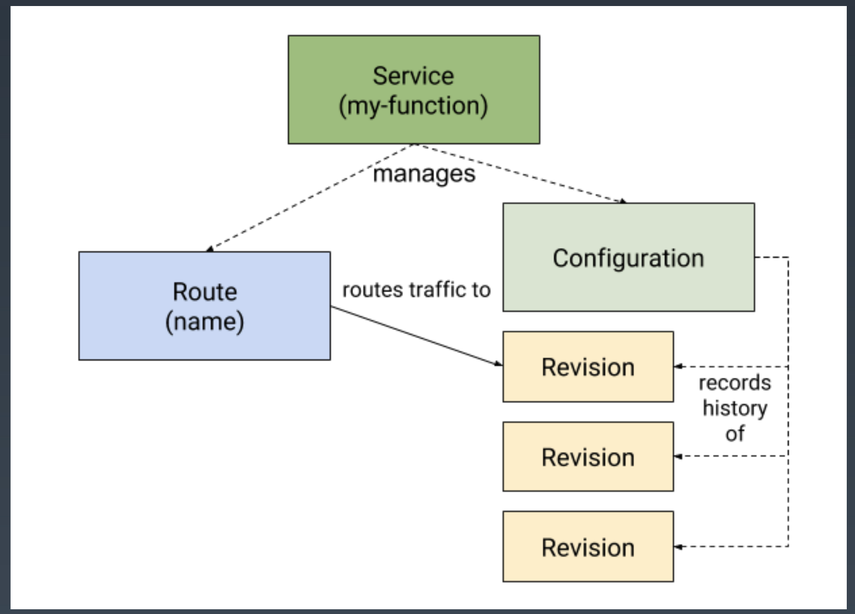


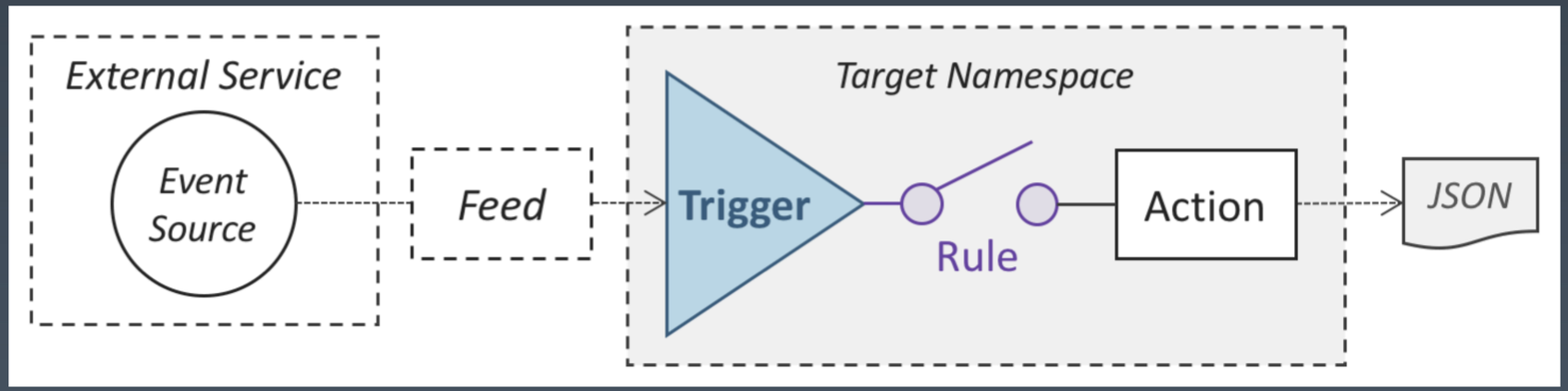
Knative 构建函数运算服务平台

三大功能：

- Build
- Serving
- Eventing

Serverless广泛应用





迷思之三： Java是否适应云原生?

1. 函数化
2. 启动速度
3. 容器镜像大小
4. 占用内存大小

FP in Java 8

```
Supplier<T>          () -> T
Function<T,R>        (T) -> R
BiFunction<T,R,Q>    (T, R) -> Q
Predicate<T>         (T) -> boolean
Consumer<T>          (T) -> {}
```

提升启动速度

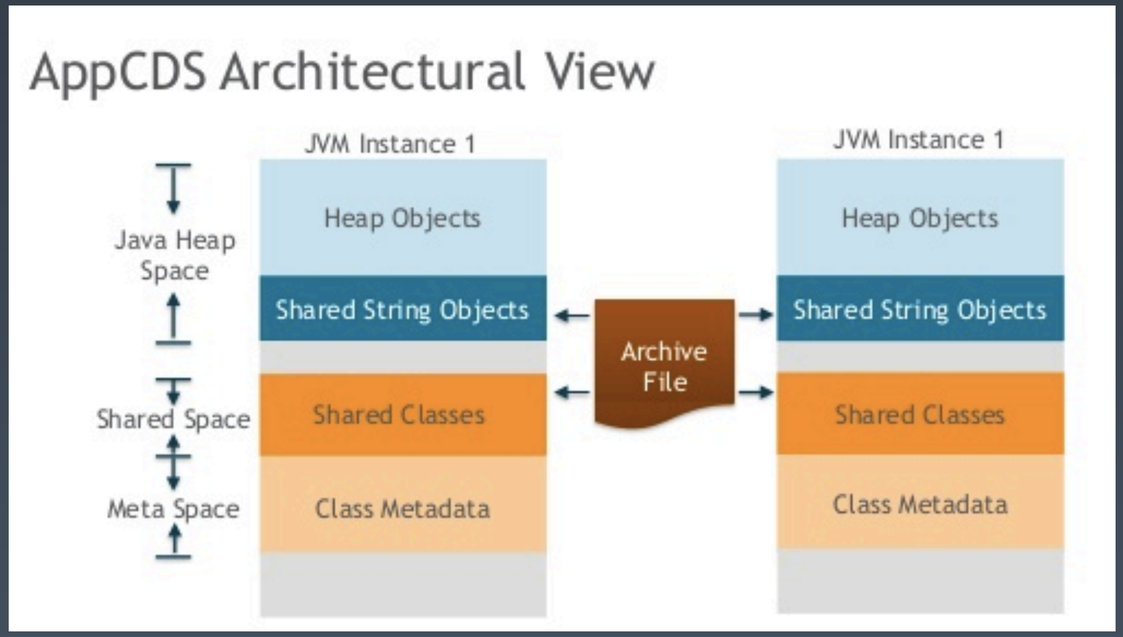
JDK AppCDS (Application Class-Data Sharing) 特性

- -Xshare:on
- -XX:+UseAppCDS

SpringBoot on JDK11

AppCDS	Startup time
off	2160ms
on	1980ms

启动速度提升8.3%。一般应用启动速度能快20%左右



裁减JDK

- Java 9 之后支持模块化
- 可以通过 Jlink 工具，只提取必须的 jmod

JDK　11	JDK占用空间	Modules文件
完整版	310.6M	130.8M
只含有 java.base mod	49.7M	24.6M

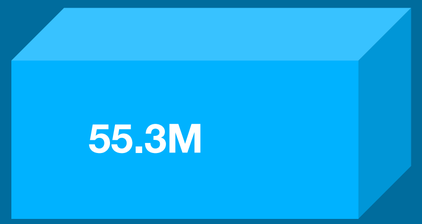
```
$ jar --create –file myapp-1.0.jar --main-class app.StringHash --module-version 1.0 -C mods .
$ jlink --module-path jmods:mods --add-modules myapp--output target
```

使用更小的镜像源

官方的OpenJDK Slim 镜像，基于Debian Slim

```
FROM debian:stretch-slim
```

Debian还是通用的Linux发行版



可以使用Alpine Linux

```
FROM alpine:3.9
```

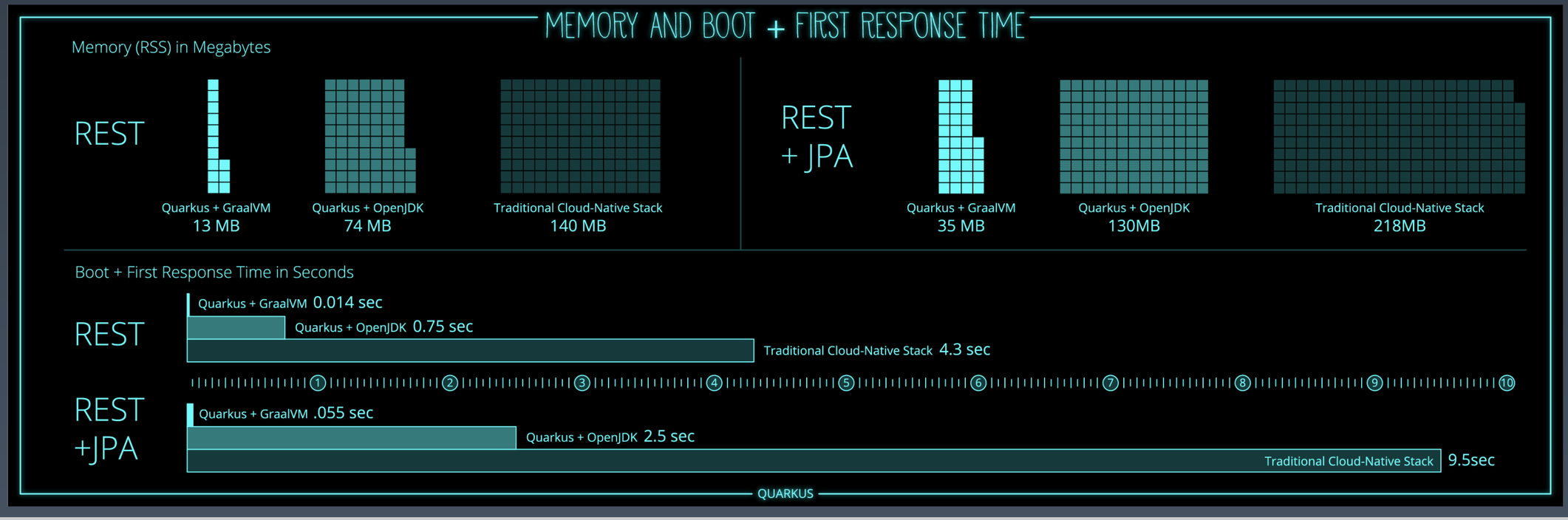
Musl-libc 和 Busybox

如何减少运行时内存占用?

- 优化 Java 应用类加载
- Java 程序“静态”化，“函数”化

Quarkus 项目

（官方给出的性能比较图，分别为 传统Java技术栈，Quarkus Java 和 Quarkus Native）



性能比较

REST数据服务

方式	Spring Boot jar包	Quarkus JVM	Quarkus native
大小	16.7M	49.5K	20M
含libs总体积	16.7M	9.8M	20M
启动时长	1850ms	530ms	10ms
占用内存(RSS)	264M	117M	13.7M
说明	SpringBoot 2.1.4, Tomcat 9	Quarkus 0.14, JDK 8	Quarkus 0.14

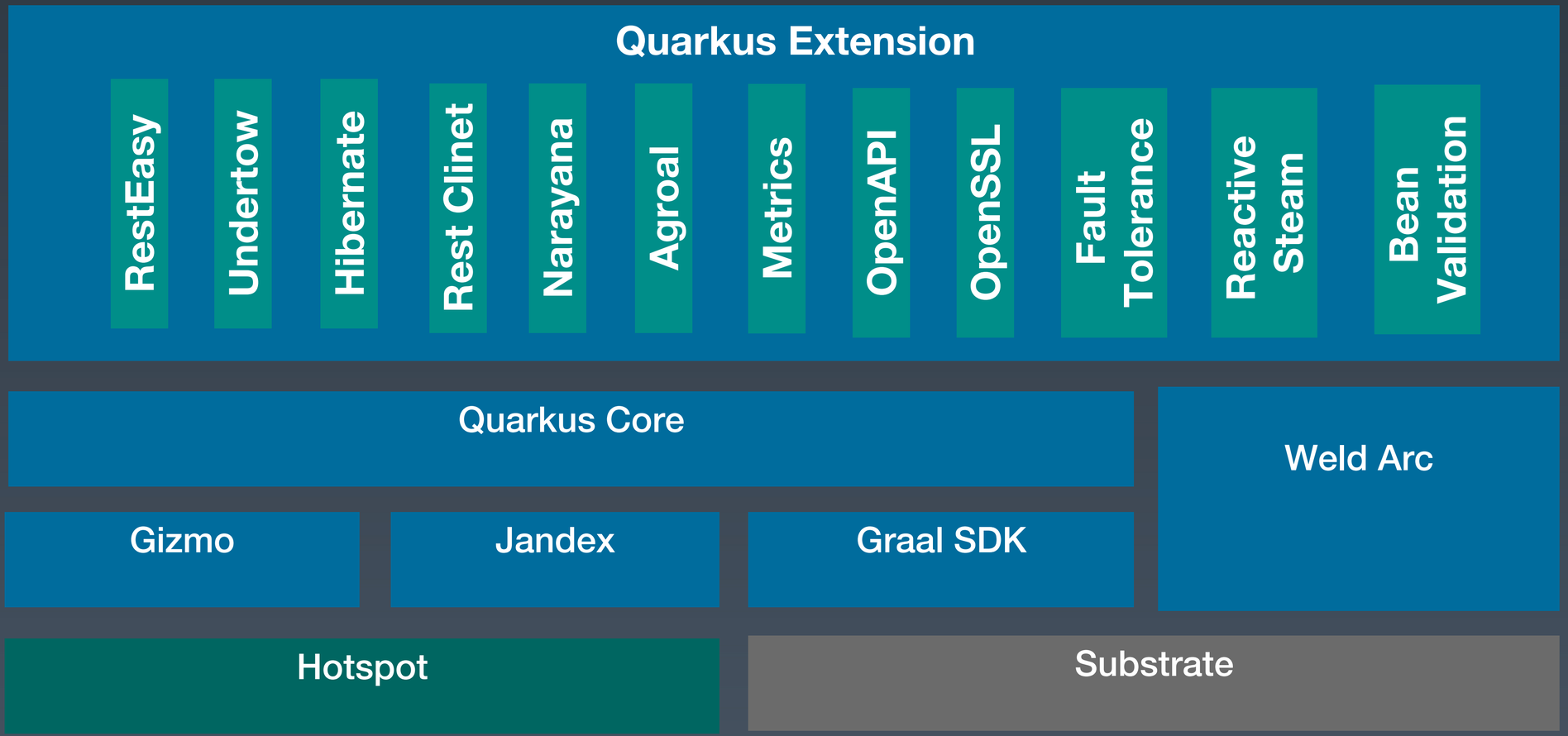
Quarkus规范化技术

借用了 Java EE 和 Microprofile 部分规范

Java EE	MicroProfile	Other
CDI	Config	OpenAPI
JAX-RS	Rest-Client	Reactive Stream
JPA	Metrics	Reactive Messaging
JSON-P / JSON-B	Fault Tolerance	KeyCloak
Servlet	JWT Security	Camel
Bean Validator		Vertx

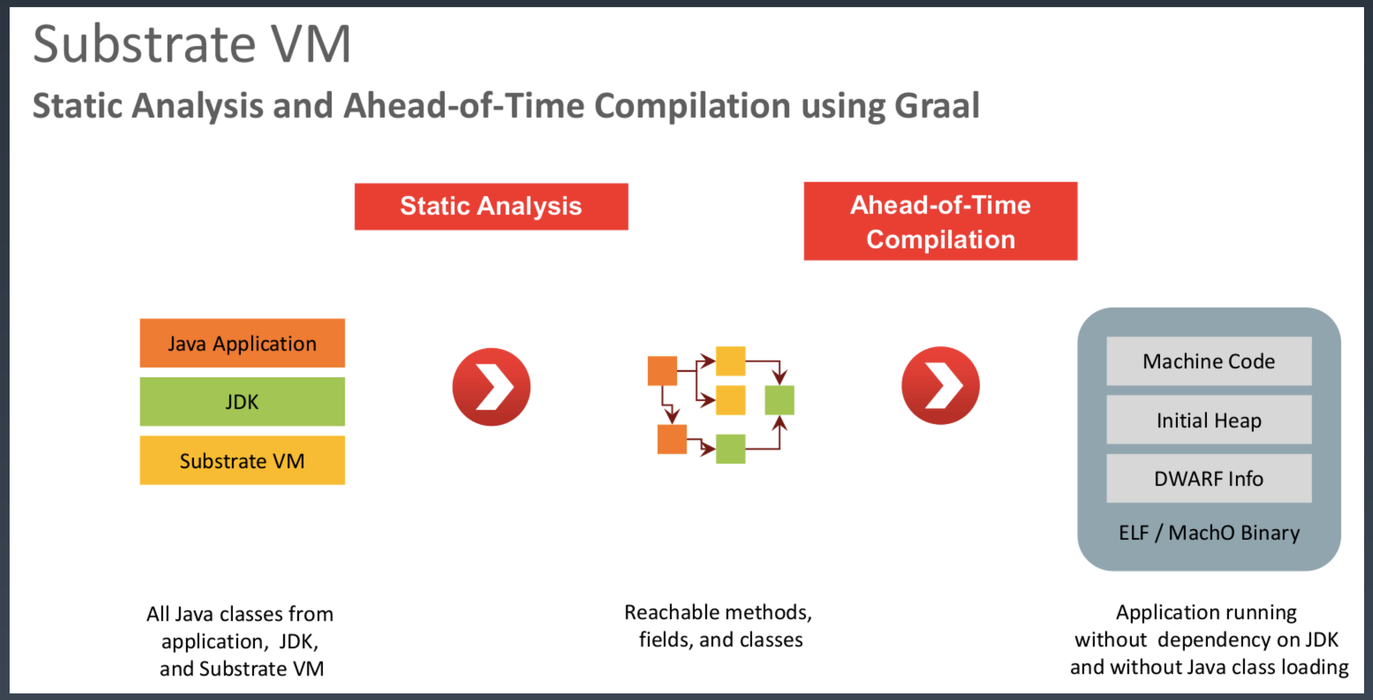
Quarkus技术栈

核心组件和主要部件



Java 原生程序

- 利用 Graal 项目的 Substrate VM中 native-image 工具
- 将 Java 应用“转换”为本地代码



编译 Compile → 提供 Provision → 组合 Augment → JDK 虚拟机运行

编译 Compile → 提供 Provision → 组合 Augment → AOT 本地编译 → 运行容器镜像

Quarkus 核心技术

- 在编译时重写 Java 代码，使得程序能够被快速加载和启动运行
- 能静态初始化加载的尽量静态构造，对原有的动态发现的进行必要代码重写
- 需要考虑本地编译的要求

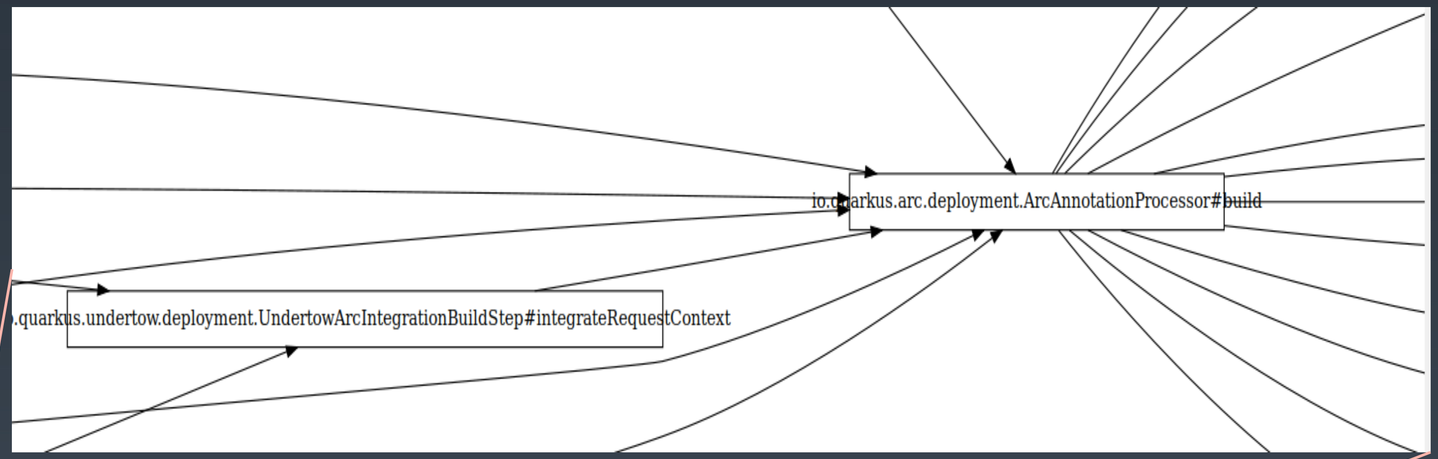
```
@AutomaticFeature
public class AutoFeature implements Feature {

public void beforeAnalysis(BeforeAnalysisAccess var1)
```

```
[INFO] --- quarkus-maven-plugin:0.14.0:native-image (default) @ using-
websockets ---
[INFO] [io.quarkus.creator.phase.nativeimage.NativeImagePhase] Running
Quarkus native-image plugin on Java HotSpot(TM) 64-Bit Server VM
[INFO] [io.quarkus.creator.phase.nativeimage.NativeImagePhase] /x1/java/
javalib/graalvm-ce-1.0.0-rc16/bin/native-image -J-
Djava.util.logging.manager=org.jboss.logmanager.LogManager -
H:InitialCollectionPolicy=com.oracle.svm.core.genscavenge.CollectionPolicy
$BySpaceAndTime -jar app-runner.jar -J-
Djava.util.concurrent.ForkJoinPool.common.parallelism=1 -
H:FallbackThreshold=0 -H:+PrintAnalysisCallTree -H:-AddAllCharsets -
H:EnableURLProtocols=http -H:NativeLinkerOption=-no-pie -H:-SpawnIsolates -
H:-JNI --no-server -H:-UseServiceLoaderFeature -H:+StackTrace
[app-runner:22276]    classlist:   2,884.84 ms
[app-runner:22276]        (cap):   1,091.63 ms
[app-runner:22276]        setup:   2,310.69 ms
20:48:44,880 WARN  [io.und.web.jsr] UT026010: Buffer pool was not set on
WebSocketDeploymentInfo, the default pool will be used
20:48:44,894 INFO  [io.und.web.jsr] UT026003: Adding annotated server
endpoint class org.acme.websocket.ChatSocket for path /chat/{username}
20:48:45,773 INFO  [org.jbo.threads] JBoss Threads version 3.0.0.Alpha4
20:48:46,336 INFO  [org.xnio] XNIO version 3.7.0.Final
20:48:46,370 INFO  [org.xni.nio] XNIO NIO Implementation Version 3.7.0.Final
[app-runner:22276]   (typeflow):  10,598.82 ms
[app-runner:22276]    (objects):   9,051.49 ms
[app-runner:22276]   (features):     817.12 ms
[app-runner:22276]    analysis:  20,971.70 ms
[app-runner:22276]    universe:     555.74 ms
[app-runner:22276]     (parse):   1,794.73 ms
[app-runner:22276]    (inline):   3,270.11 ms
[app-runner:22276]   (compile):  19,523.92 ms
[app-runner:22276]     compile:  25,859.57 ms
[app-runner:22276]       image:   2,127.83 ms
[app-runner:22276]       write:     437.76 ms
[app-runner:22276]     [total]:  57,628.60 ms
```

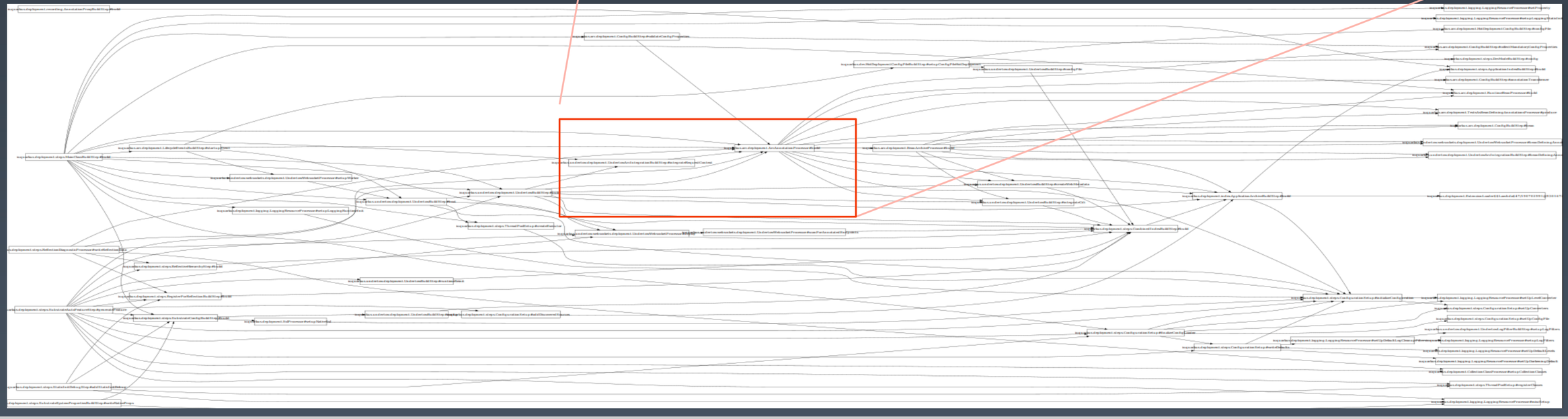
Quarkus Buildsteps

内存中构建 Build Step 过程图



全局视图

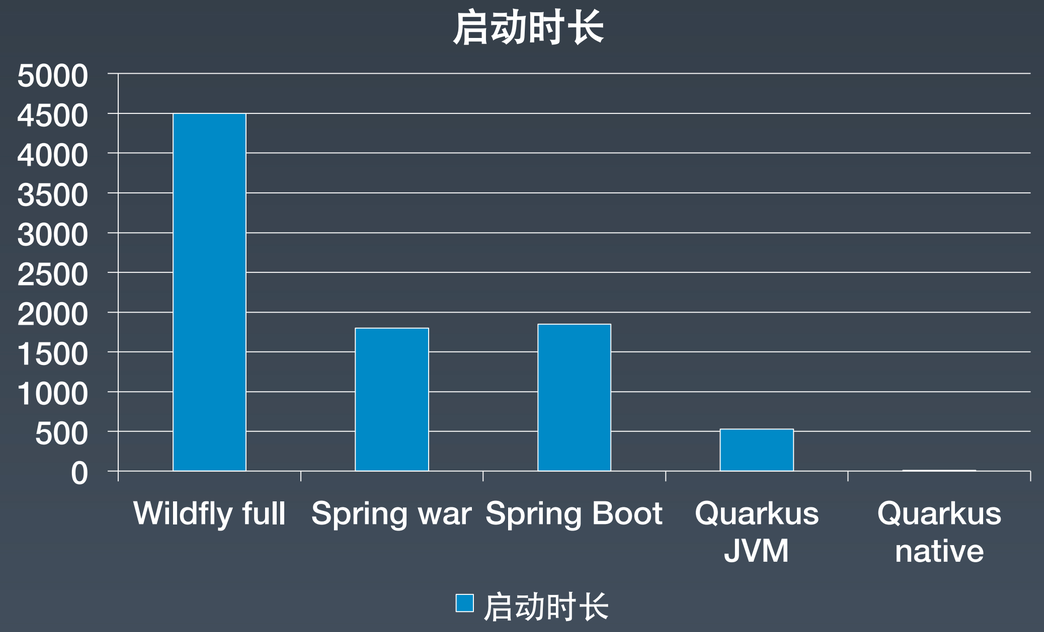
+ 局部放大

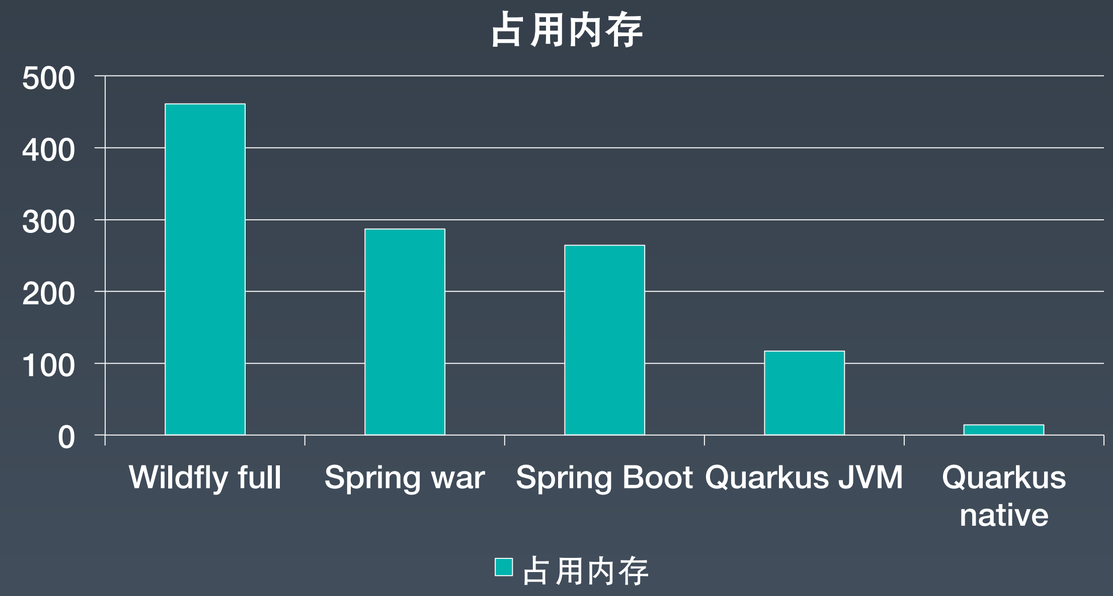


容器化部署图表

启动时长和占用内存(RSS)

	Wildfly Full	Spring War	Spring Boot	Quarkus JVM	Quarkus Native
启动时长(ms)	4500	1800	1850	530	10
占用内存(MB)	461	287	264	117	14





容器化部署技术选择

适合的才是最好的

- 用户规模，访问频次
- 组件状态和事务要求
- 运维团队和技能
- 云计算设施情况
- Java 和其他语言以及相关技术栈的熟悉程度

THANKS! | QCon 10th