

Cloud Pak for Applications Warsztaty Praktyczne

Mikołaj Jaworski

Partner Technical Specialist @ IBM

Grzegorz Smółko

Hybrid Cloud Solutions Design @ IBM

Rafał Owczarek

Customer Success Manager Architect @ IBM

Paula Rutkowska

Automation Portfolio Sales @ IBM



Agenda

8:45 - 8:50: Rejestracja i przypisanie stanowisk roboczych.

8:50 - 9:00: Rozpoczęcie

- Powitanie i krótka prezentacja prowadzących
- Przedstawienie agendy, celów i oczekiwań od warsztatów

9:00 - 10:20: Wykład 1: Nowoczesne aplikacje wymagają nowoczesnych narzędzi - modernizacja aplikacji z Cloud Pak for Applications

- Co napędza modernizację aplikacji i dlaczego jest ważna?
- Różne strategię do modernizacji aplikacji - Plan modernizacji.
- CP4Apps - narzędzia wspierające modernizację.
- 6 powodów, dla których warto wybrać Liberty!

10:20 - 10:45: DEMO 1: Narzędzia deweloperskie dla Liberty na przykładzie VSC

- Wykorzystanie narzędzi OpenLiberty Tools w VSC
- Doświadczenie programisty w korzystaniu z narzędzi Open Liberty Tools w VS Code
- Uruchamianie testów przy użyciu narzędzi Open Liberty w VS Code

10:45 - 11:00: Przerwa na kawę

11:00 - 11:30: Wykład 2: Modernizacja aplikacji z IBM Transformation Advisor

- Jak narzędzia modernizacyjne mogą pomóc?
- Jak działa IBM Transformation Advisor?
- Analiza aplikacji - Raporty TA.

11:30 - 12:30: Lab 1: TA: zbieranie danych, ocena aplikacji oraz akceleracja wdrożenia do Liberty

- Uruchamianie TA.
- Analiza aplikacji Java w TA dla różnych środowisk docelowych.
- Wykorzystanie akceleratorów TA, aby wdrożyć i uruchomić aplikację "Mod Resorts" w Open Liberty.

12:30 - 13:15: Lunch

13:15 - 13:30: Wykład 3: Modernizacja aplikacji z Mono2Micro

- Jak działa IBM Mono2Micro?
- IBM Mono2Micro - Rekomendacje mikrouług.
- Automatyczne generowanie kodu.

13:30 - 13:45: DEMO 2: Modernizacja aplikacji z Mono2Micro

- Wykorzystanie Mono2Micro do analizy aplikacji monolitycznej Java EE podział na partycje.

13:45 - 15:00: Wykład 4: Wstęp do Red Hat OpenShift Container Platform

- RHOC - komponenty.
- RHOC - korzyści.

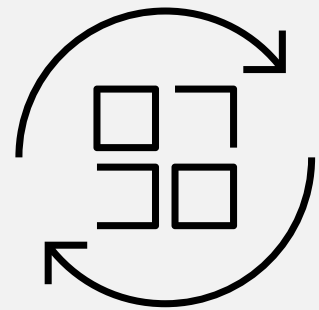
15:00 - 15:15: Przerwa na kawę

15:15 - 16:30: Lab 2: Wdrażanie aplikacji na OpenShift przy użyciu Open Liberty Operator

- Modernizacja środowiska wykonawczego.
- Modernizacja operacyjna - wdrożenie aplikacji na OpenShift.

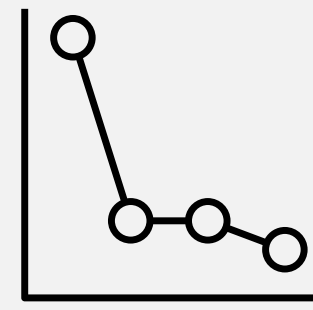
16:30 – 17:00 Podsumowanie i zakończenie spotkania

Co napędza modernizację?



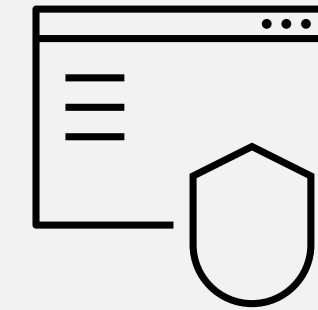
Speed-to-market

Przyjęcie nowoczesnych praktyk i standardów oprogramowania w celu skrócenia czasu wprowadzania produktów na rynek.



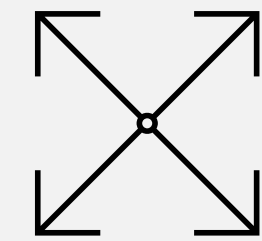
Redukcja kosztów

Zmniejszenie wymagań dotyczących infrastruktury aplikacji i wymagań obliczeniowych oraz osiągnięcie celów zrównoważonego rozwoju i optymalizacji kosztów.



Ograniczenie ryzyka

Bezpieczny cyfrowy łańcuch dostaw i ograniczenie narastania długu technologicznego.



Dostęp do innowacji w oprogramowaniu

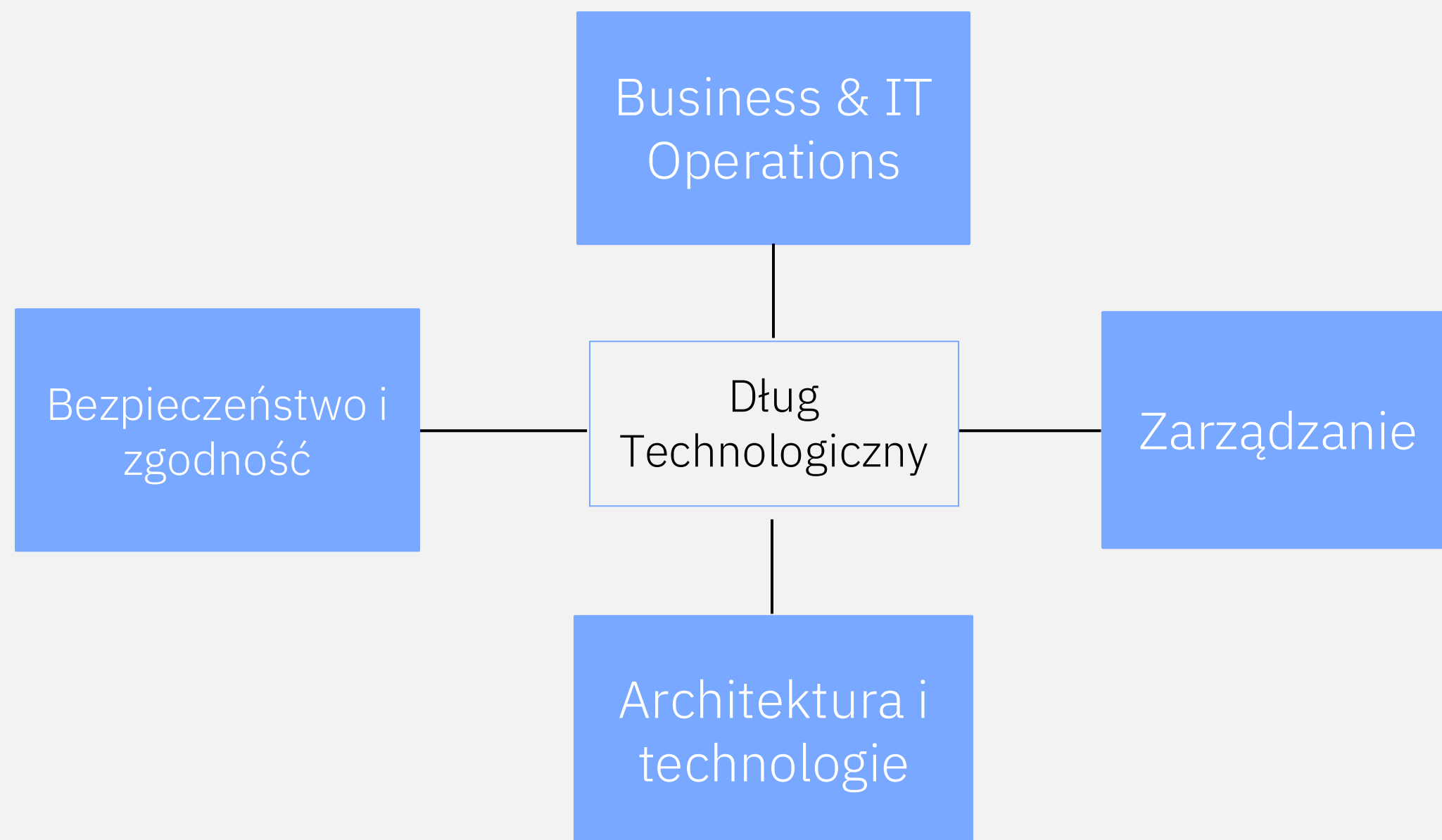
Wykorzystanie nowych technologii w celu poprawy wyników biznesowych.

Dług technologiczny jest główną przeszkodą w osiągnięciu celów biznesowych

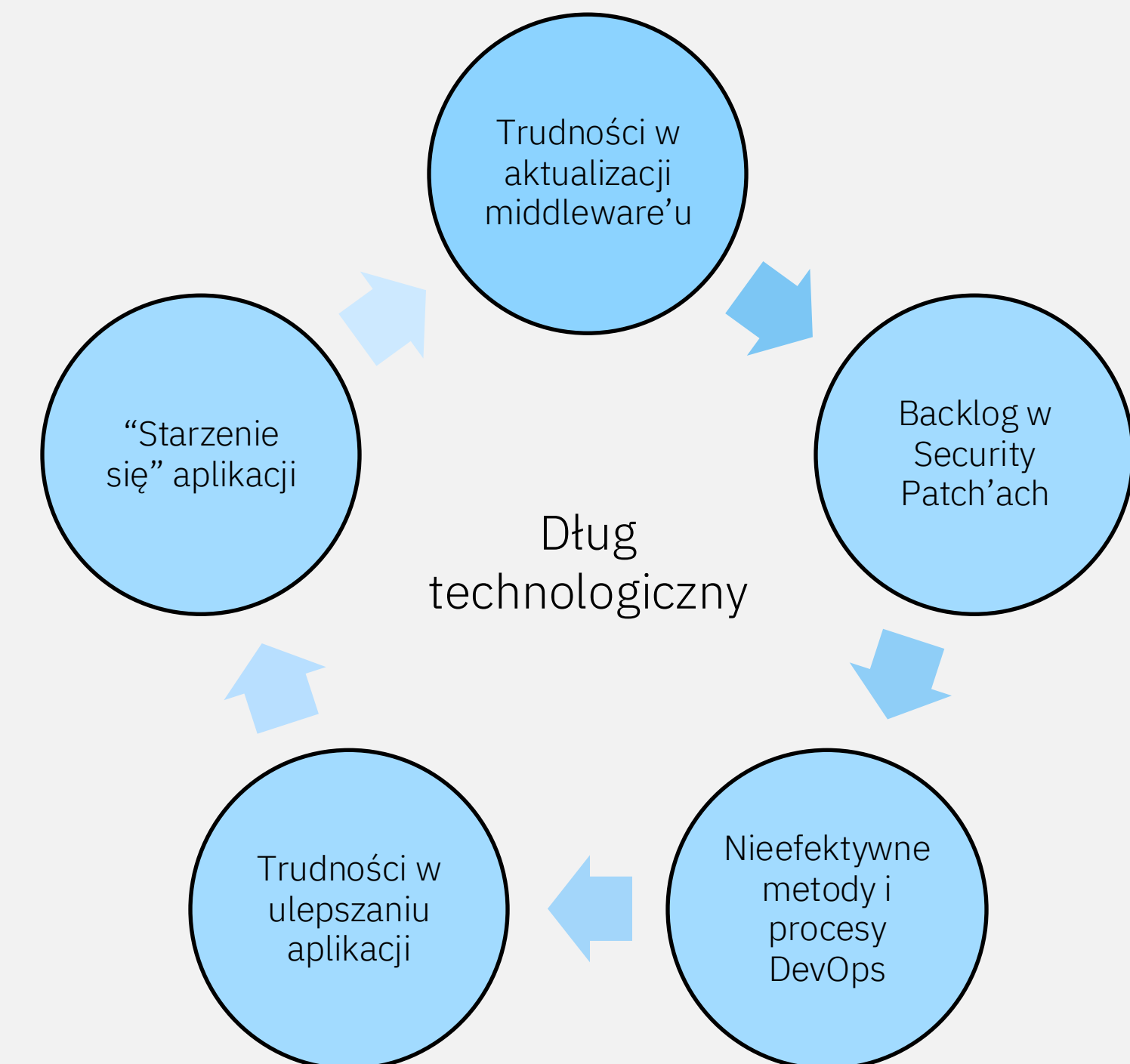
Co to jest?

„Dług technologiczny to naturalne zjawisko, polegające na tym, że koszt dodania identycznej funkcjonalności do danego projektu IT wzrasta wraz z rozwojem projektu IT”²

Kim są współautorzy:



Dług technologiczny dotyczy aż **40%** zasobów technologicznych¹

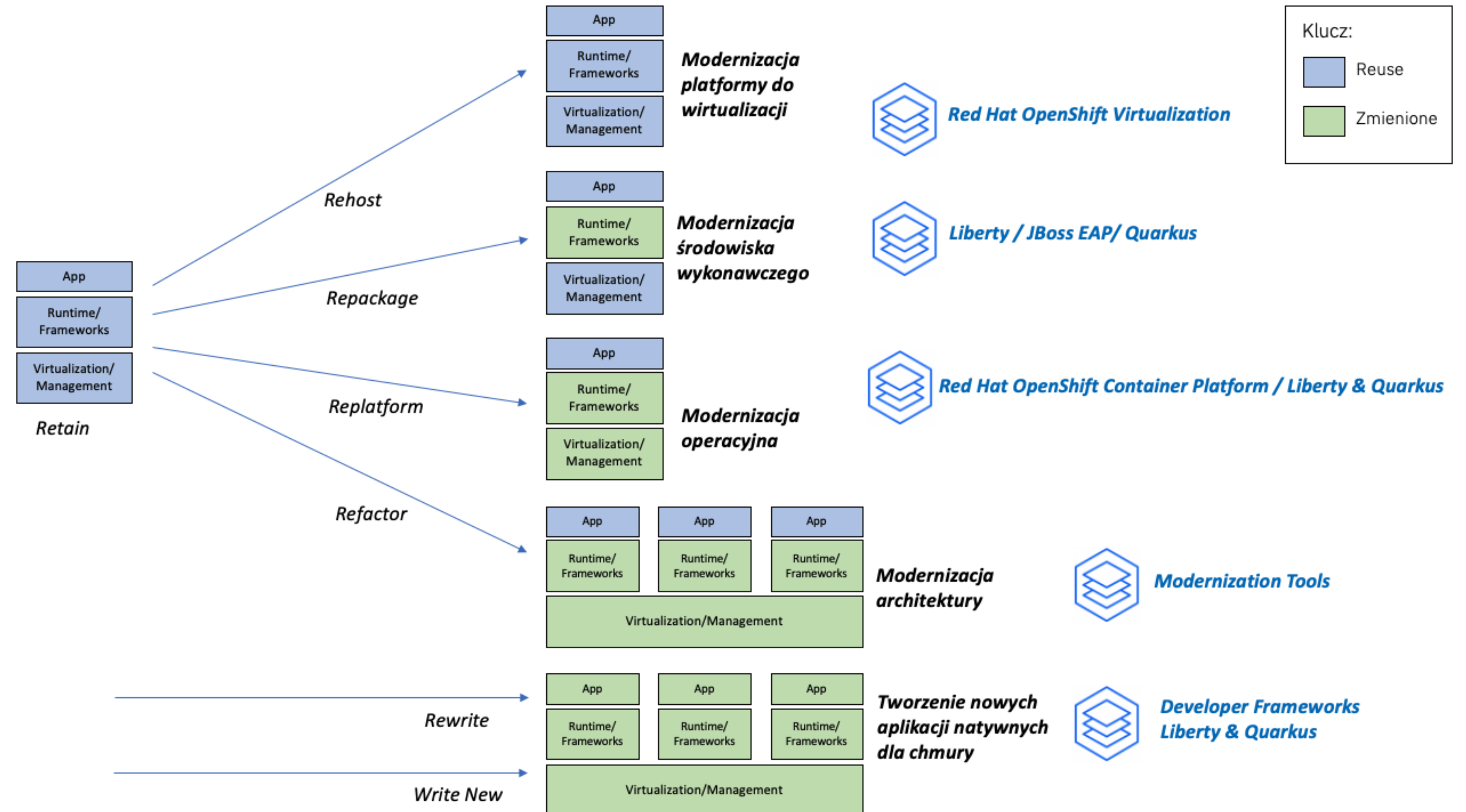


Podejście do modernizacji aplikacji z CP4Apps

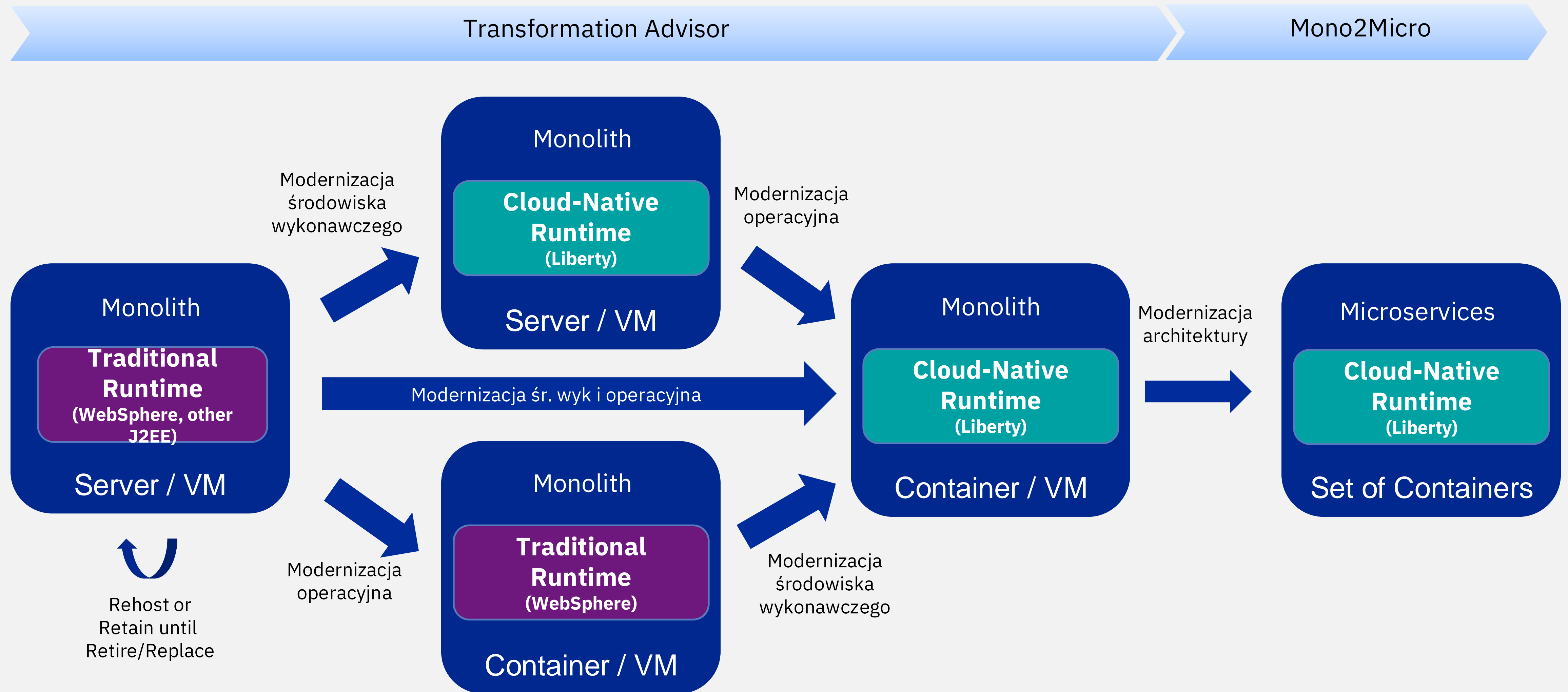
Czynniki wpływające na modernizację:

- Cele biznesowe
- Czas
- Koszt
- Możliwości organizacyjne
- Wydajność
- Platforma
- Umiejętności
- Oceny ryzyka
- Zależności od stron trzecich
- Itp..

Nie ma uniwersalnej odpowiedzi



Jaki jest plan na modernizację?



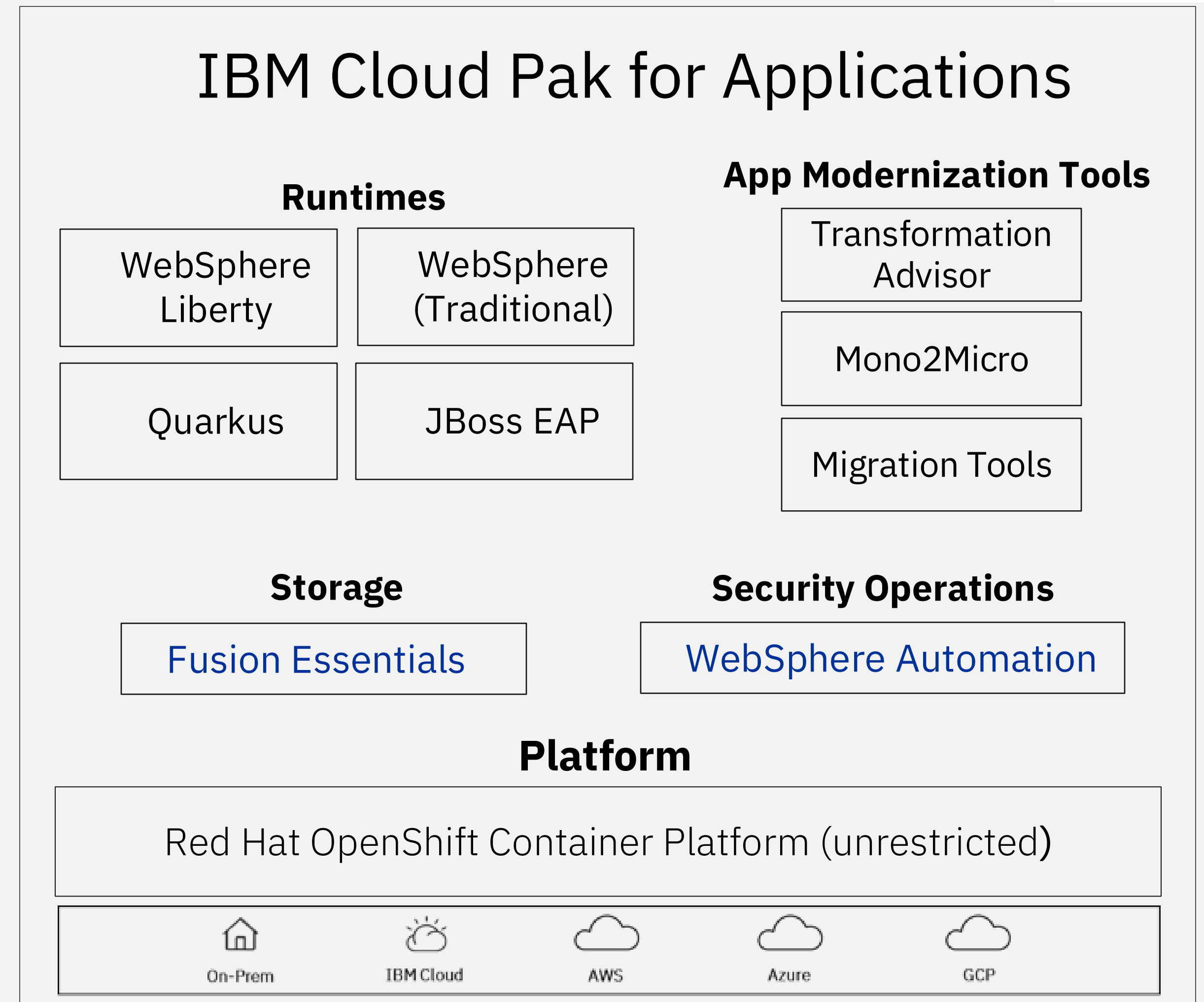
Czym jest IBM Cloud Pak for Applications?

Dostarcz rozwiązanie aplikacyjne, które rozwija się wraz z Twoim biznesem.



Wybierz to czego potrzebujesz:

- ✓ Kompleksowe środowiska uruchomieniowe obsługujące maszyny wirtualne i kontenery dla aplikacji Java.
- ✓ Narzędzia modernizacyjne do refaktoryzacji i dekompozycji aplikacji na potrzeby wdrożenia w chmurze.
- ✓ Kompleksowa platforma oparta na Kubernetes obsługująca cały krajobraz aplikacji w środowisku lokalnym i poza nim.
- ✓ Zautomatyzowane operacje bezpieczeństwa dla wybranych środowisk wykonawczych.
- ✓ Dedykowany storage dla platformy kontenerowej.



Cloud Pak for Applications

Pełny wybór opcji dla chmury hybrydowej z możliwością elastycznego łączenia i dopasowywania narzędzi

IBM Runtimes

- Open Liberty
- WebSphere Application Server
 - WAS (base)
 - WAS Liberty (base)
- WebSphere Application Server Liberty Core
- WebSphere Application Server Network Deployment
 - WAS ND
 - WAS Liberty ND

IBM Modernization Tools

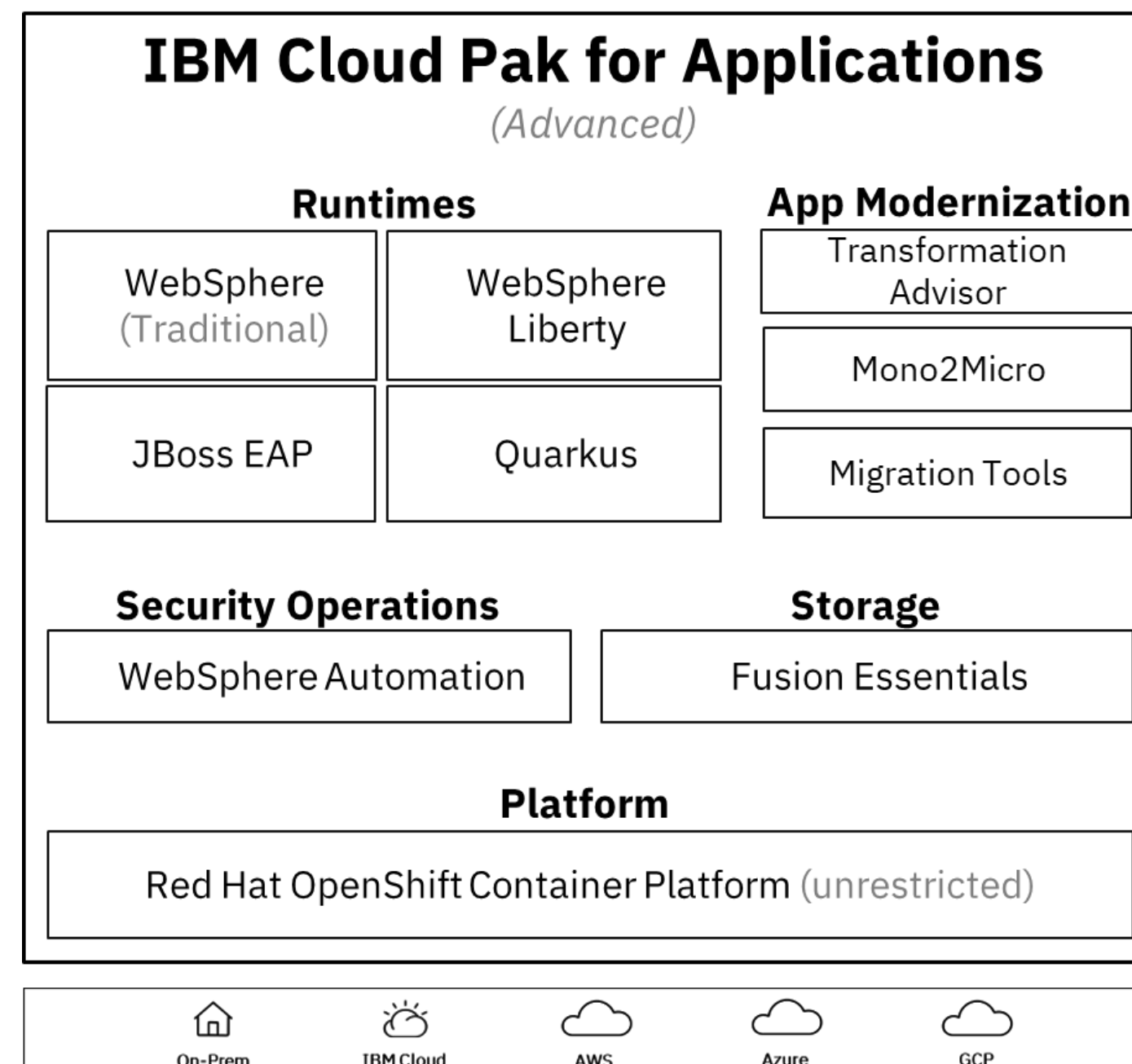
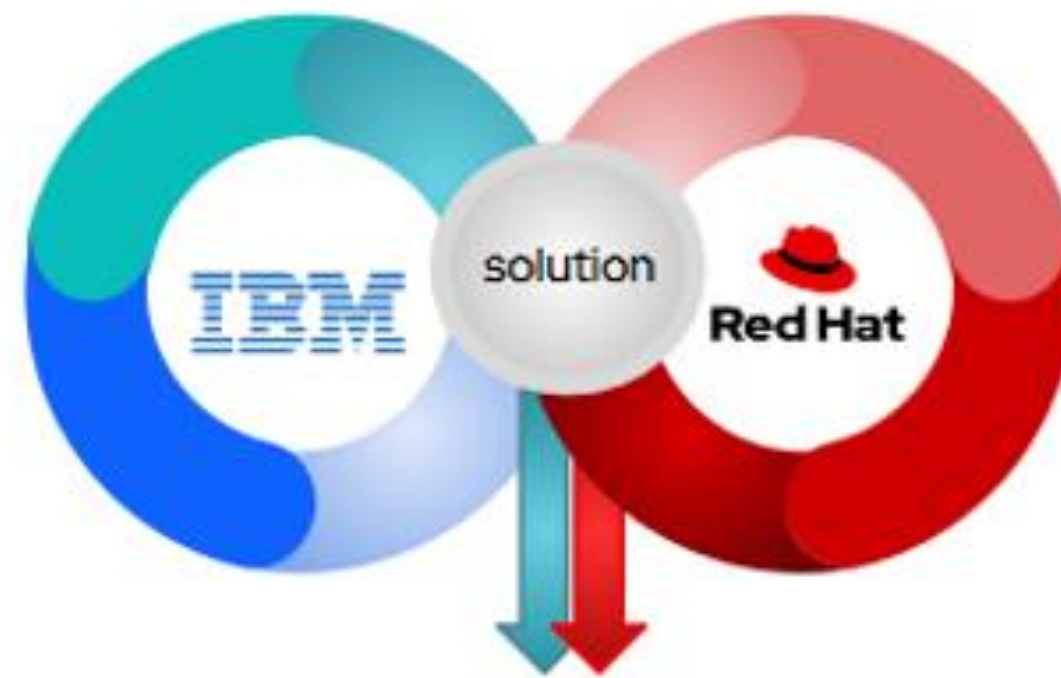
- Mono2Micro
- Transformation Advisor
- WebSphere Application Server Migration Toolkit

IBM Security Operations

- IBM WebSphere Automation

IBM Storage

- IBM Storage Fusion Essentials



Red Hat OpenShift Container Platform

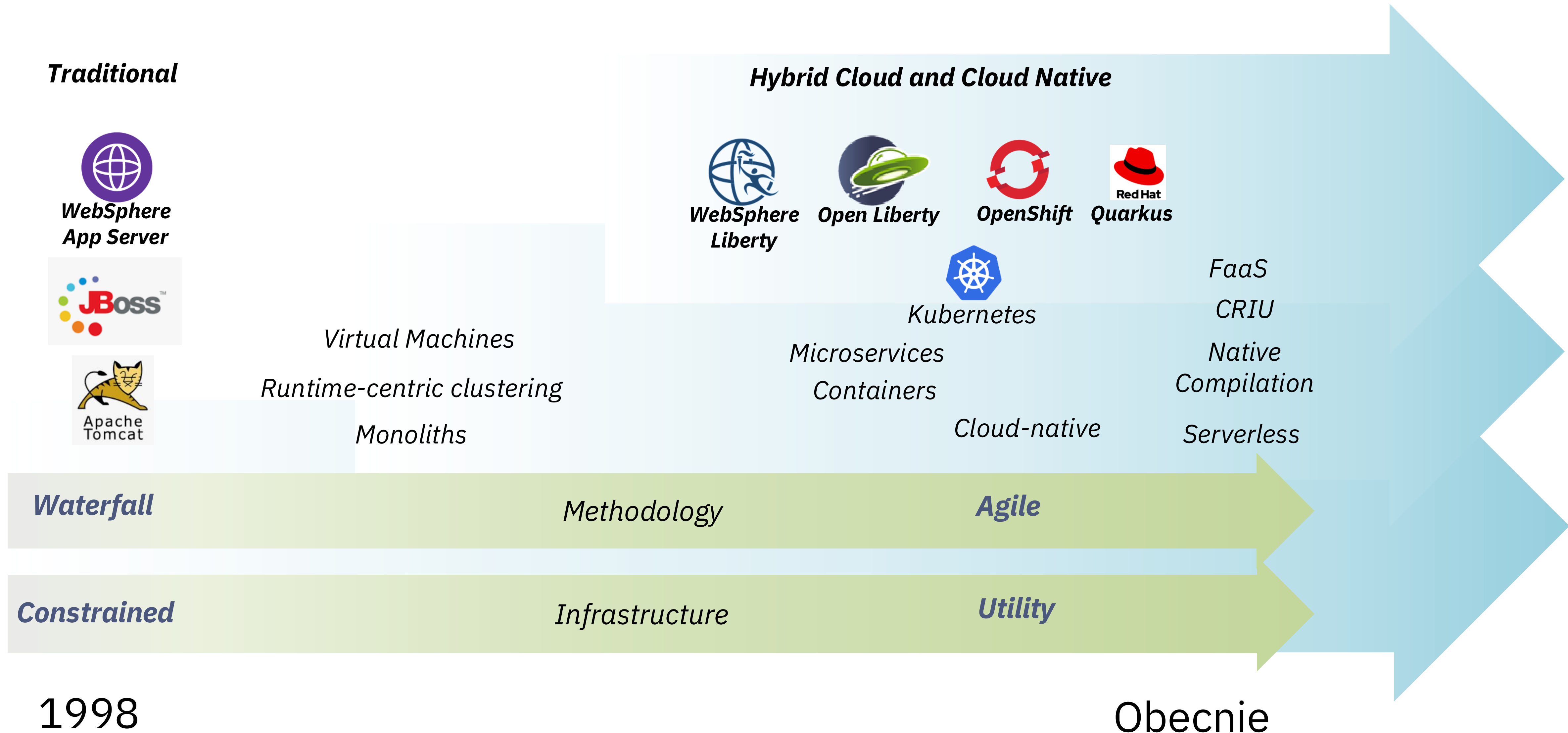
- Migration Toolkit for Virtualization
- Red Hat Enterprise Linux
- Red Hat Enterprise Linux CoreOS
- Red Hat JBoss Web Server for OpenShift
- Red Hat OpenShift Developer console
- Red Hat OpenShift GitOps
- Red Hat OpenShift Kubernetes Engine
- Red Hat OpenShift Pipelines
- Red Hat OpenShift Serverless
- Red Hat OpenShift Service Mesh
- Red Hat OpenShift Virtualization

Red Hat Runtimes

- JBoss Enterprise Application Platform
- JBoss Web Server
- Migration Toolkit for Applications
- Red Hat AMQ
- Red Hat build of Keycloak
- Red Hat build of OpenJDK
- Red Hat build of Quarkus
- Red Hat Data Grid

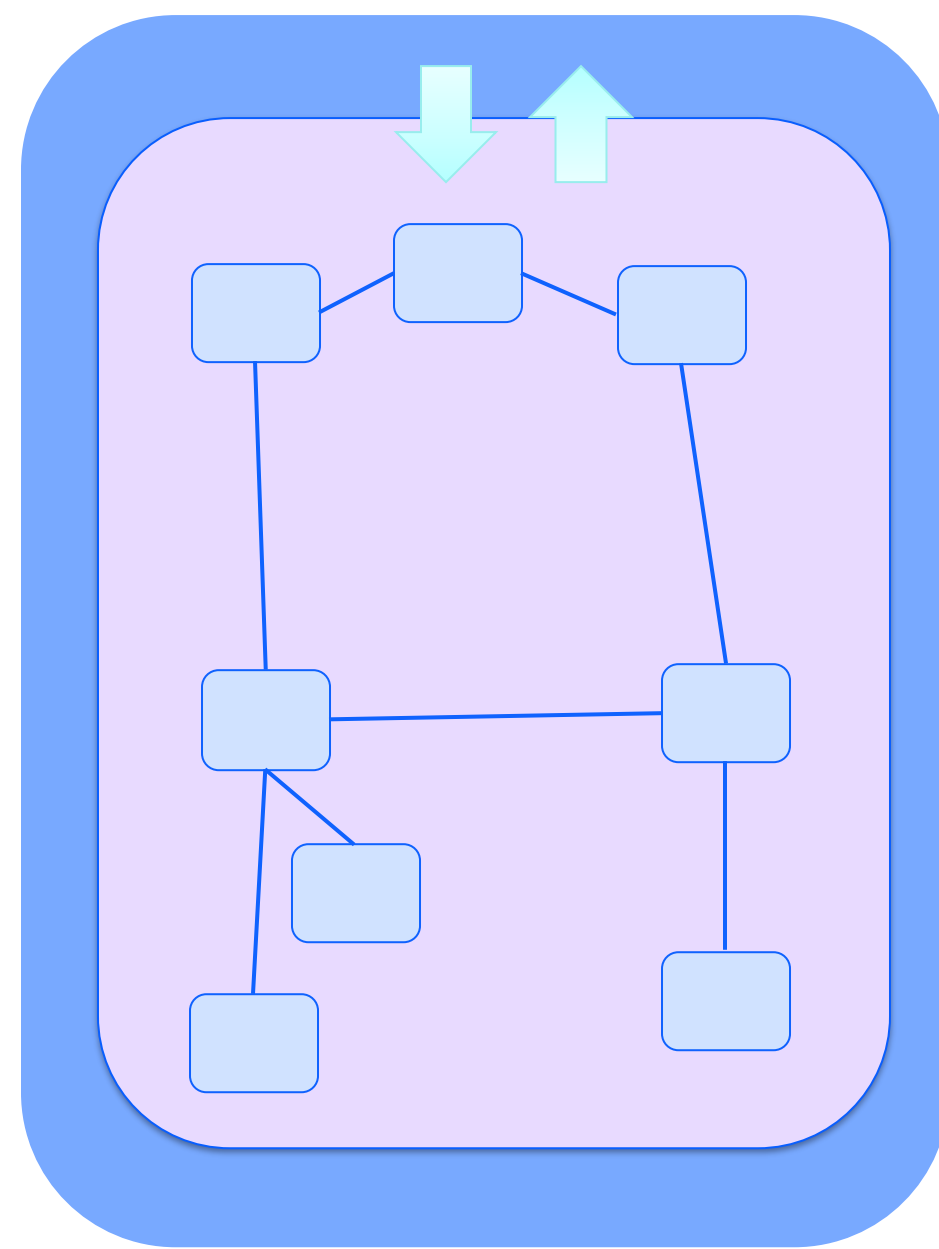
Java jest jednym z najpopularniejszych języków programowania aplikacji

- Obsługuje architekturę monolityczną, makroserwisy i mikroserwisy ...

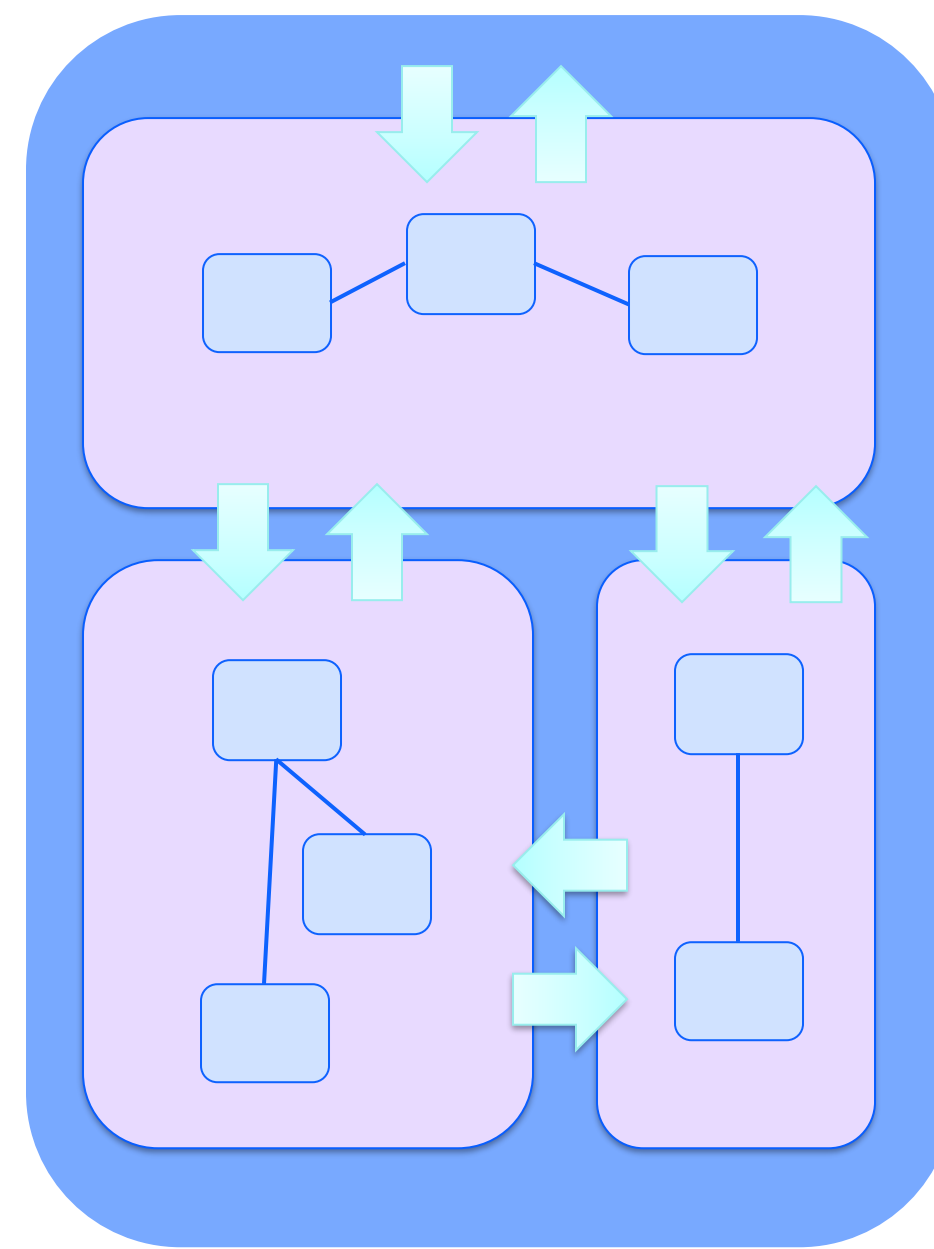


Liberty to środowisko uruchomieniowe IBM dla modernizacji Enterprise Java i nowych środowisk wykonawczych.

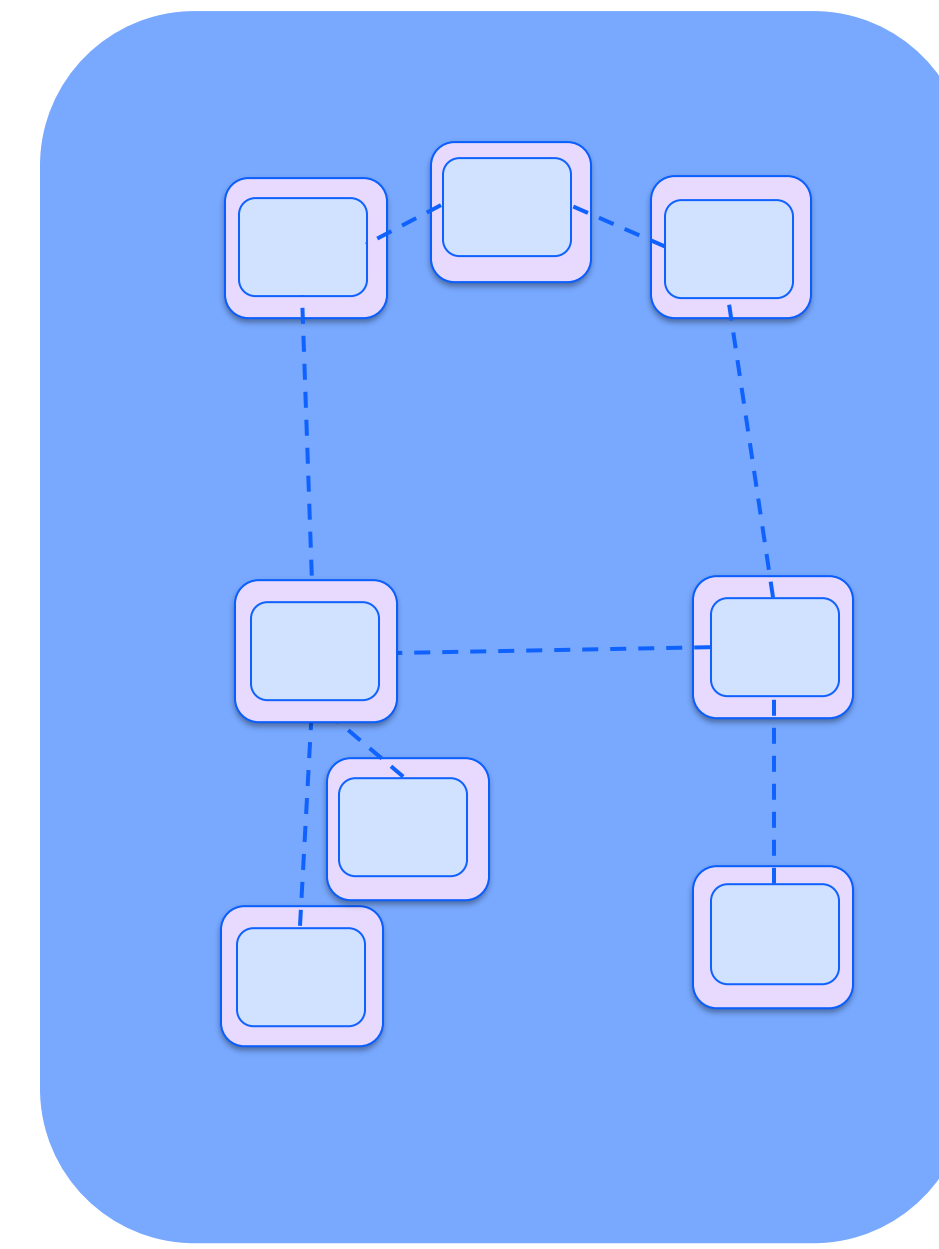
Zaprojektowany dla maszyn wirtualnych i kontenerów, Java EE, Jakarta EE, MicroProfile, Spring Boot, monolitów i mikroustług



Monolit



“Makroserwisy”



Mikroserwisy



Modernizacja środowiska wykonawczego do Liberty minimalizuje dług technologiczny!

- 6 powodów dlaczego warto wybrać Liberty: <https://developer.ibm.com/articles/6-reasons-why-open-liberty-is-an-ideal-choice-for-developing-and-deploying-microservices/>

Poprawa zrównoważonego rozwoju i niższe koszty infrastruktury

1. Runtime taki jak potrzebujesz



80% oszczędności dysku i 56% pamięci

2. Niższe koszty operacyjne



4x większa gęstość w porównaniu do Tomcat i Spring Boot

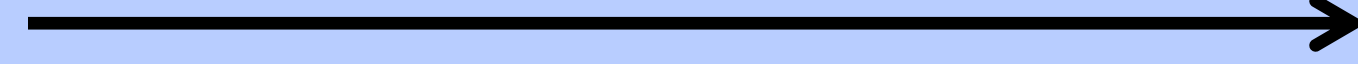
Poprawa bezpieczeństwa i niższe koszty administracyjne

3. Ciągłe dostarczanie



Zero nakładów na poprawki bezpieczeństwa i zero długu technologicznego

4. Zero migracji



100% oszczędność migracji v2v i fixpack

Zwiększa produktywność deweloperów

5. Optymalne dla Kubernetes



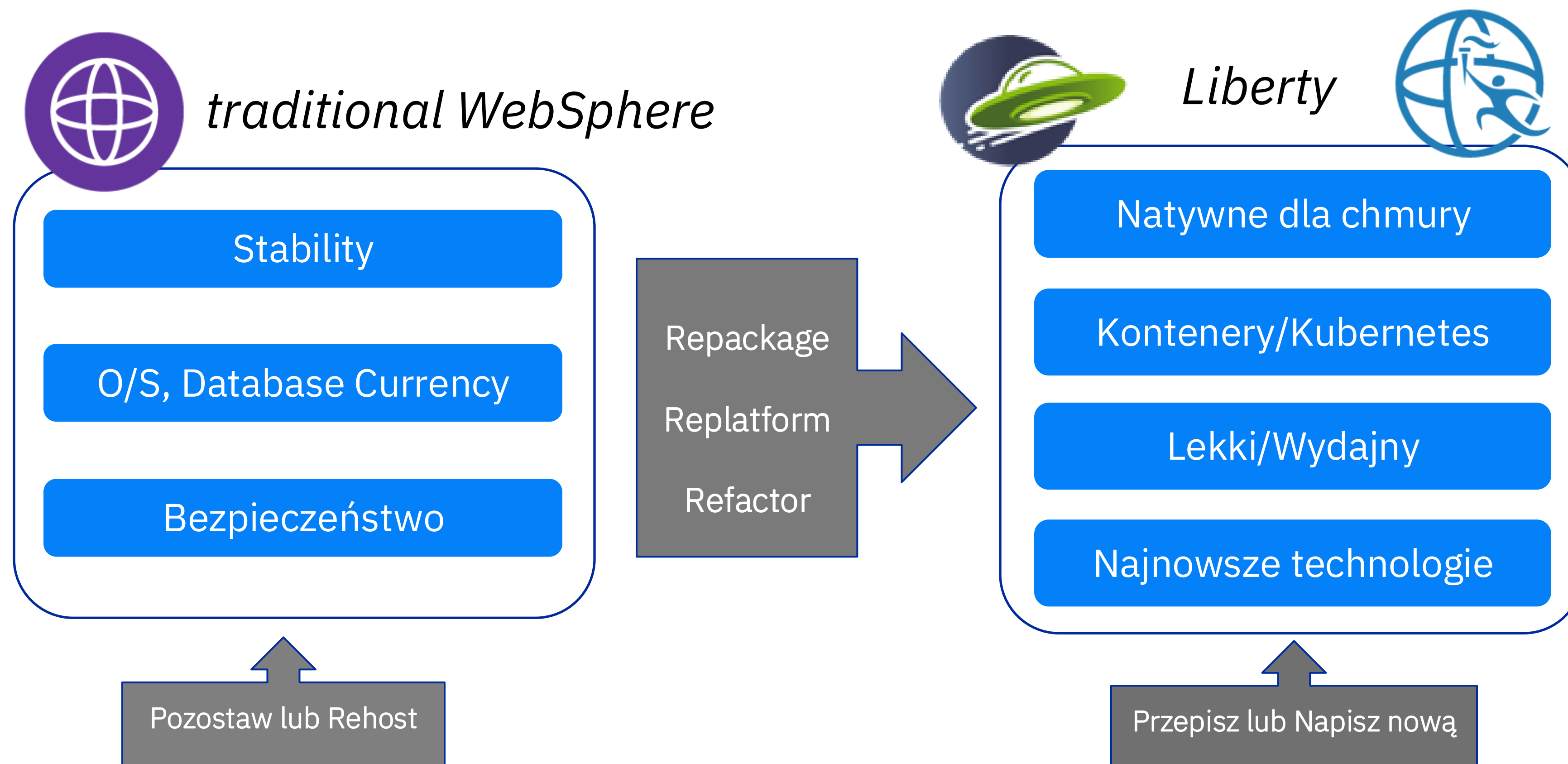
Samodzielnie dostrojona optymalna wydajność, gotowa do produkcji, natywna dla kube

6. Optymalne dla programistów



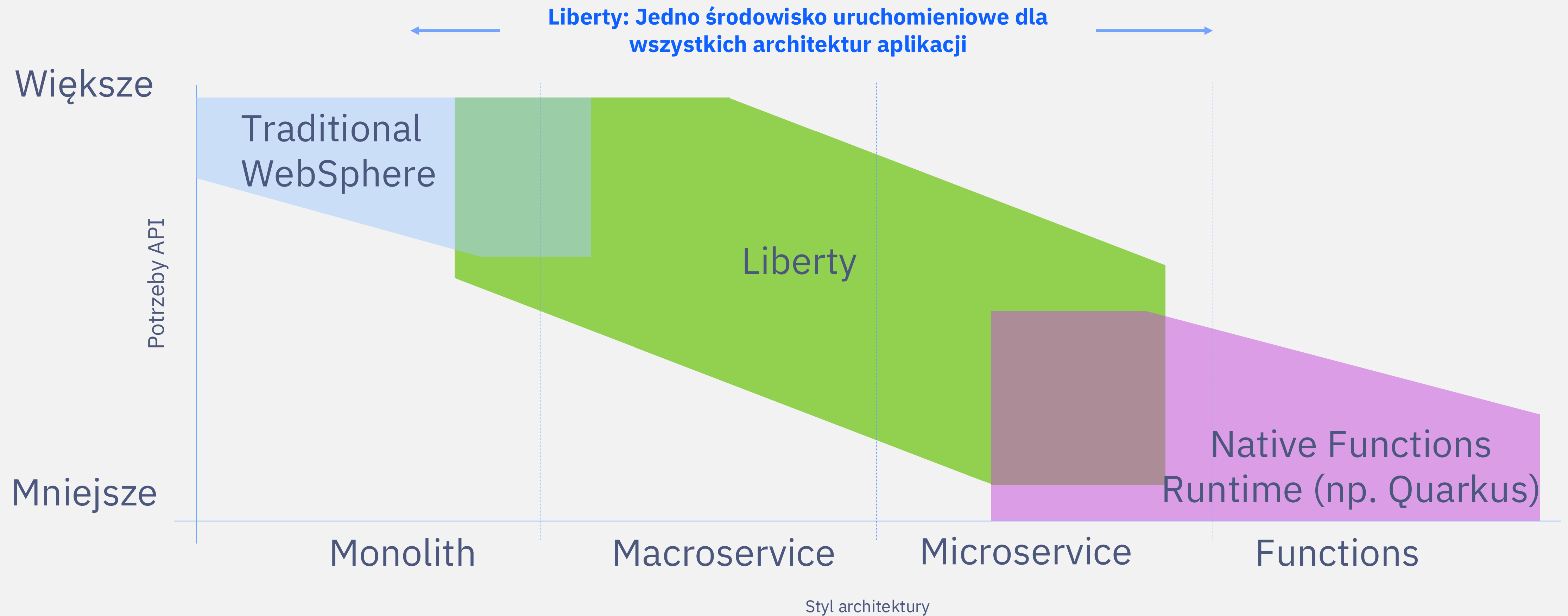
Szybkie tworzenie, testowanie i debugowanie

Wsparcie obecnych i przyszłych potrzeb w zakresie środowiska wykonawczego poprzez modernizację z WebSphere do Liberty



Odpowiednie środowisko wykonawcze Java

Potrzeby API dla różnych stylów architektury



Open Liberty

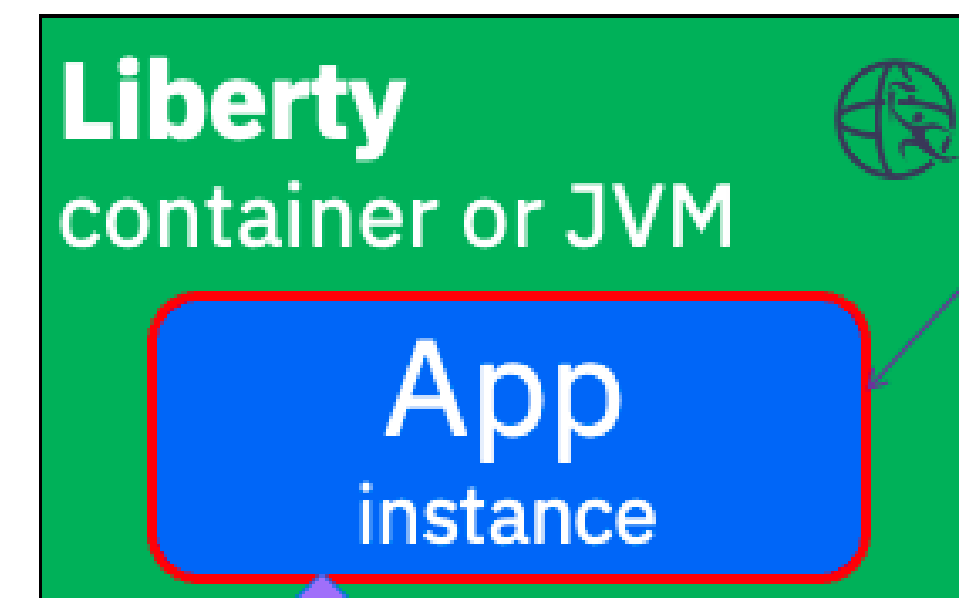
Zaprojektowany z myślą zarówno o programistach, jak i właścicielach aplikacji.

- Liberty dostarcza najnowsze interfejsy API Java i integruje się z najpopularniejszymi narzędziami deweloperskimi.
- Liberty ma wbudowane technologie pozwalające na redukcję kosztów uruchamiania aplikacji i nakładów związanych z ich wdrażaniem.
- WebSphere Liberty jest rozwinięciem Open Liberty, więc wszystko, co działa na Open Liberty, działa na WebSphere Liberty.
- Nie musisz przechodzić na WebSphere Liberty, aby uzyskać wsparcie komercyjne

Built on open source



<https://openliberty.io/>



**MicroProfile and
Jakarta EE APIs for
Applications**

**Liberty Tools With All
Popular Developer
Environments (IDEs)**

1. Runtime taki jak potrzebujesz

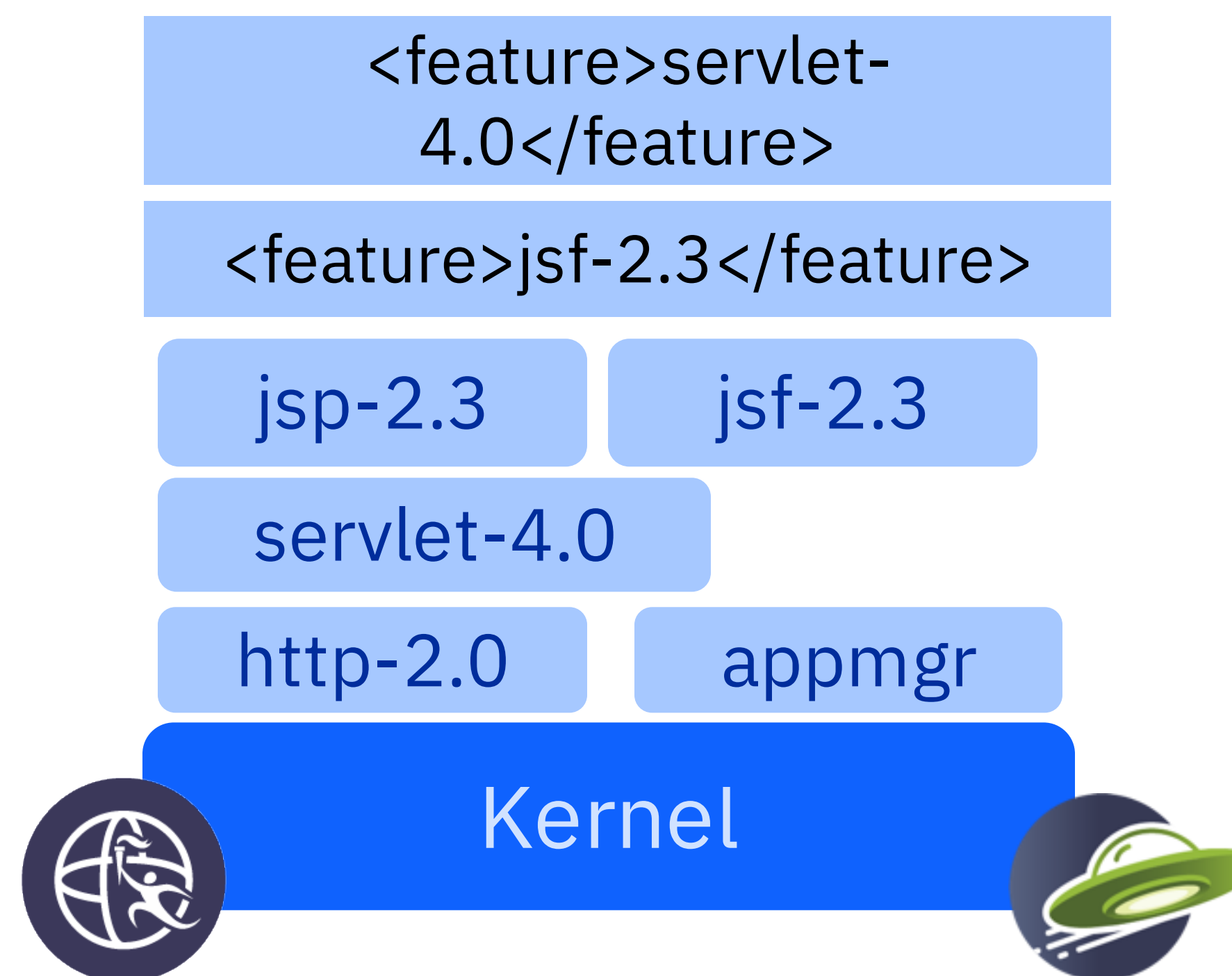
Uruchamiaj tylko to, czego wymaga Twoja aplikacja.

Zbuduj serwer jakiego potrzebujesz

Tradycyjny App Server: pełny stos API Java EE, a także funkcje administracyjne i operacyjne są ładowane w każdej instancji serwera.

Liberty: zaczynając od lekkiego jądra, kontrolujesz, które funkcje są ładowane do każdej instancji serwera.

Oszczędność dysku i pamięci na instancję oraz zmniejszenie powiązanych kosztów hostingu



Liberty - Otwarte środowisko uruchomieniowe Java Enterprise Runtime

zOS






























































ND

Base

Core

Open Liberty

as of 23.0.0.12

batchSMFLogging-1.0				zosLocalAdapters-1.0		zosTransaction-1.0			
				zosRequestLogging-1.0		zosWlm-1.0		zosSecurity-1.0	
collectiveController-1.0		dynamicRouting-1.0		healthManager-1.0		scalingController-1.0			
		clusterMember-1.0		healthAnalyzer-1.0		scalingMember-1.0			
cloudant-1.0		jakartaee-9.1		batchManagement-1.0 				acmeCA-1.0 	
javaee-8.0		jakartaee-8.0		wsAtomicTransaction-1.2 				wsSecurity-1.1 	
javaee-7.0		heritageAPIs-1.1						wsSecuritySaml-1.0 	
jakartaee-10.0		sipServlet-1.1							
appAuthorization-2.1		mpGraphQL-2.0		adminCenter-1.0		audit-1.0 		ldapRegistry-3.0 	
bells-1.0		mpReactiveMessaging-1.0		collectiveMember-1.0		constrainedDelegation-1.0 		oauth-2.0 	
facesContainer-4.0		mpReactiveStreams-1.0		distributedMap-1.0		federatedRepository-1.0 		openid-2.0 	
grpc-1.0		osgiConsole-1.0		eventLogging-1.0		jwt-1.0 		openidConnectClient-1.0 	
jdbc-4.3		persistenceContainer-3.1		logstashCollector-1.0		jwtSso-1.0 		openidConnectServer-1.0 	
json-1.0		springBoot-3.0		monitor-1.0		sessionDatabase-1.0 		passwordUtilities-1.1 	
jsonbContainer-3.0		webProfile-10.0		openapi-3.1		webCache-1.0 		samlWeb-2.0 	
jsonpContainer-2.1		webProfile-9.1		requestTiming-1.0		wmqMessagingClient-3.1		scim-1.0	
mail-2.1		webProfile-8.0		usageMetering-1.0				socialLogin-1.0 	
microProfile-6.1		webProfile-7.0		restConnector-2.0				spnego-1.0 	
mpContextPropagation-1.3		xmlBinding-4.0		sessionCache-1.0				transportSecurity-1.0 	

APIs

[For details see Liberty features - IBM Documentation](#)

Operations

Security

Liberty - Otwarte środowisko uruchomieniowe Java Enterprise Runtime



2. Niższe koszty operacyjne

Mniej pamięci, wysoka przepustowość, mniej instancji i mniej licencji

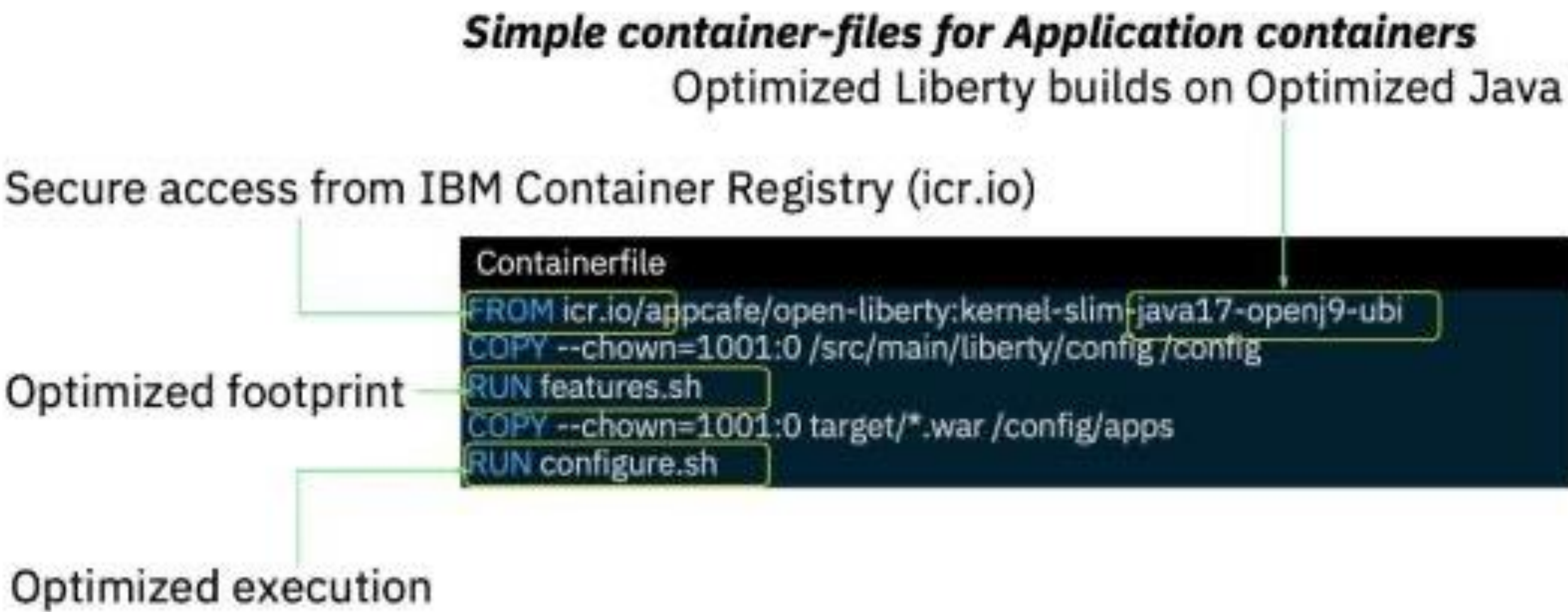
Prosta budowa w odpowiednim rozmiarze

Tworzenie aplikacji

- Wtyczki Maven i Gradle
- Wszystkie artefakty Liberty opublikowane maven central

Tworzenie kontenera

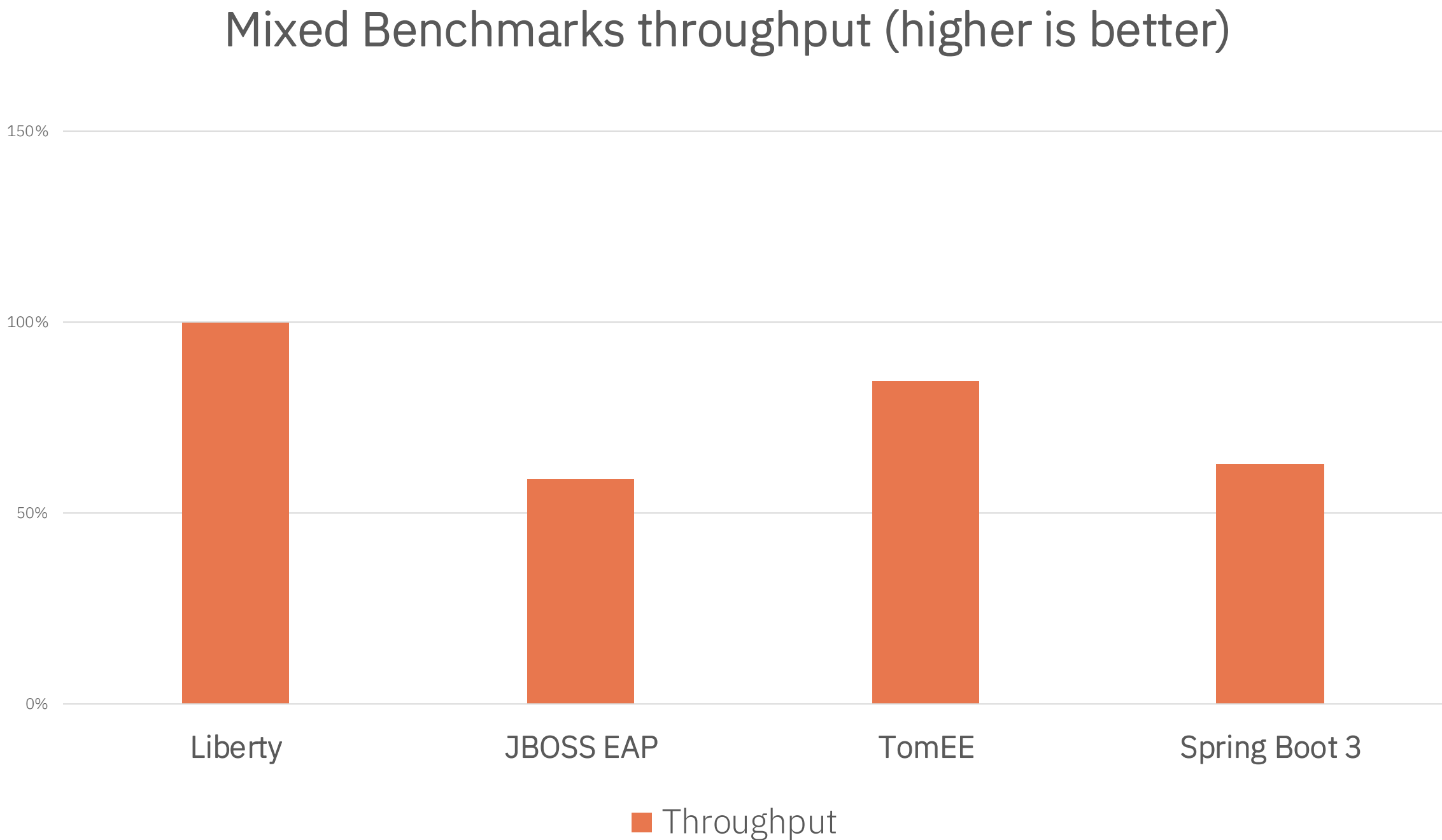
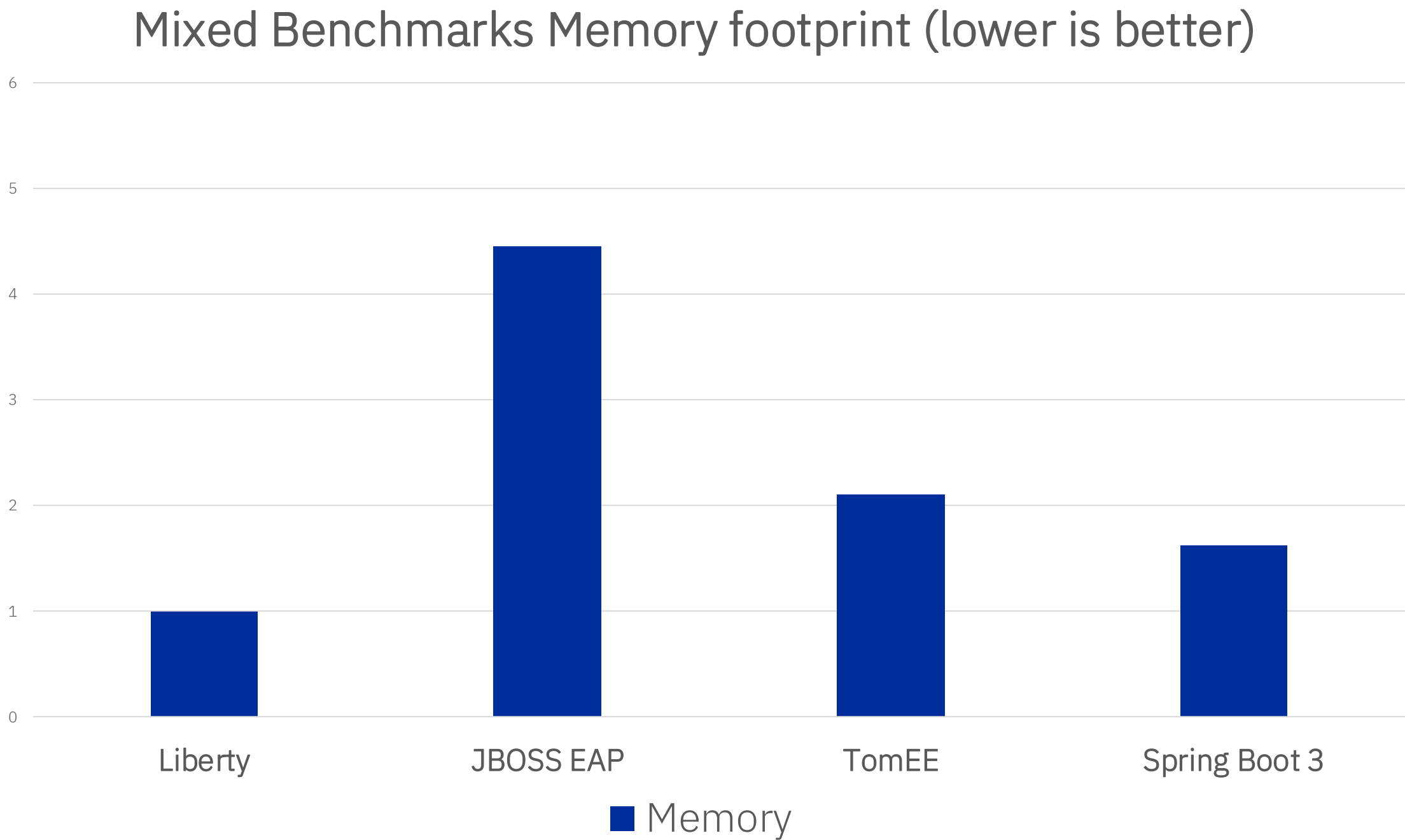
- Wiodące podejścia do tworzenia kontenerów - Dockerfile, Cloud Native Buildpack, Source-2-Image
- Certyfikowane obrazy Liberty opublikowane w IBM Container Registry



Funkcjonalność	Rozmiar na dysku	Zajętość pamięci
Java EE 8 / Jakarta EE 8 + MicroProfile 3.3	121MB	165MB
MicroProfile 3.3	59MB	113MB
Servlet 4.0	24MB	72MB

Niskie koszty operacyjne - dowolna architektura, maszyny wirtualne lub kontenery

Doskonałe wyniki Liberty są porównywane



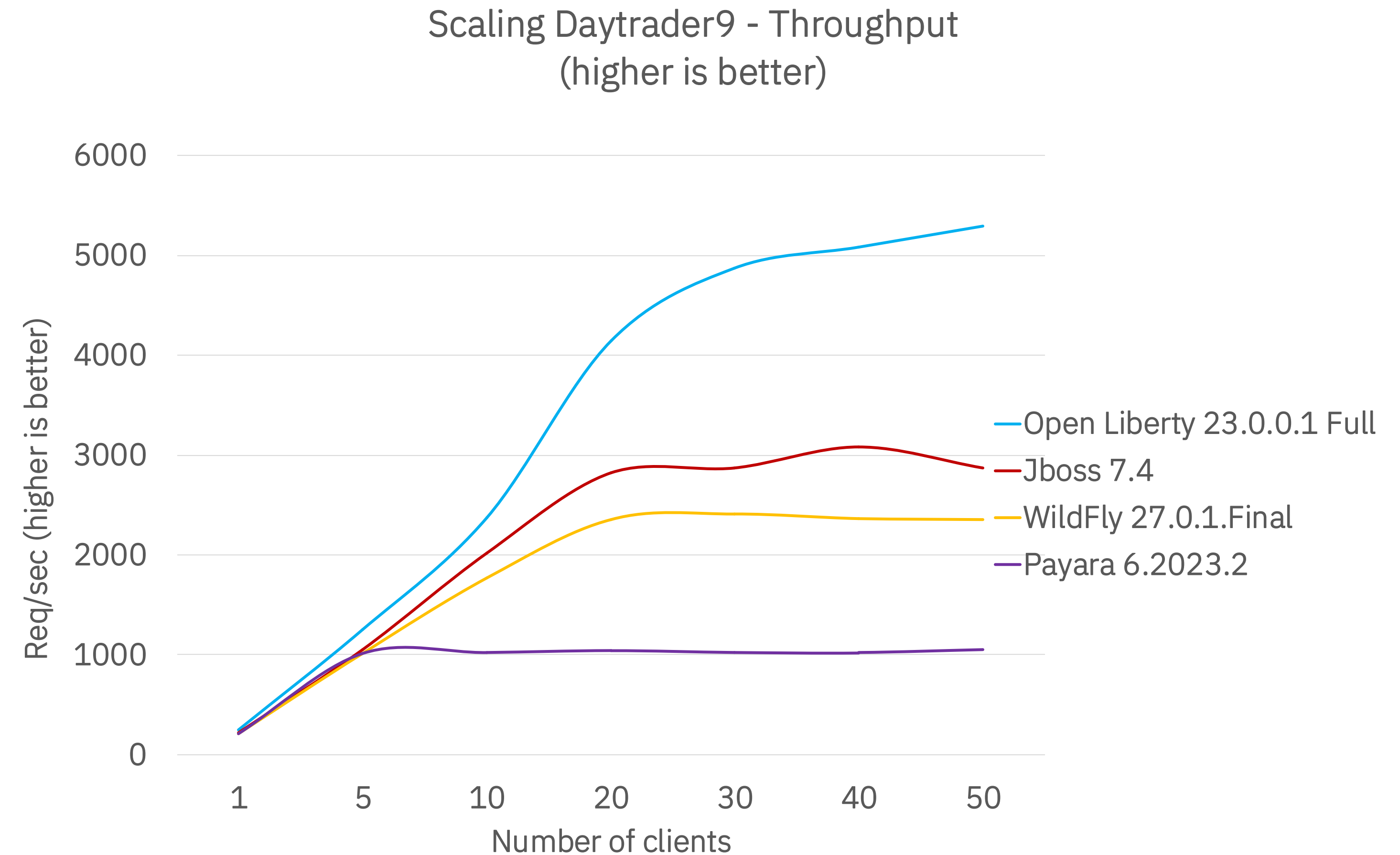
System Configuration:

SUT: LinTel – RHEL 8.7, Intel(R) Xeon(R) Gold 6338 @ 2.00GHz, 4 cpus, 4 GB RAM.
JDK 17 version distributed with the docker images used for each server instance.

- ✓ Najlepsza przepustowość
- ✓ Najlepszy ślad pamięciowy
- ✓ Najmniej licencji
- ✓ Najmniej sprzętu
- ✓ Najniższy koszt
- ✓ Najmniejszy wpływ na środowisko

Skalowanie

- Liberty skaluje się lepiej wraz ze wzrostem liczby klientów



System Configuration:

SUT: LinTel – RHEL 8.7, Intel(R) Xeon(R) Gold 6338 @ 2.00GHz, 4 cpus, 4 GB RAM.
JDK 11 version distributed with the docker images used for each server instance.

3. Ciągłe dostarczanie

Niskie koszty utrzymania, zerowy dług technologiczny, uwzględnione poprawki bezpieczeństwa

4. Zero migracji

Eliminacja problemów migracji z wersji do wersji

Ciągłe dostarczanie Liberty

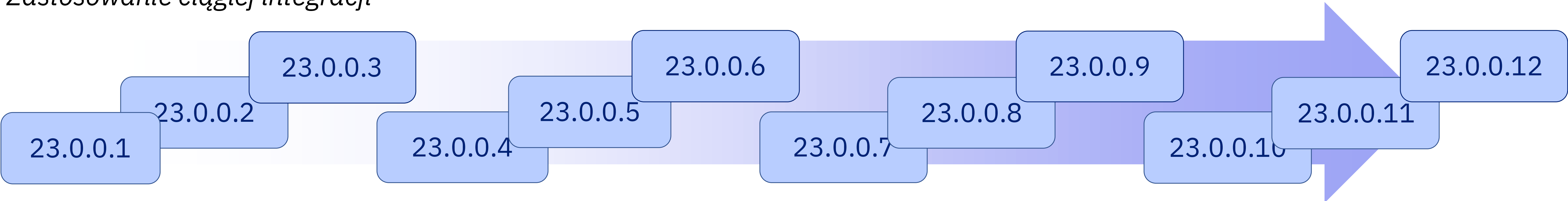
Architektura „zerowej migracji” Liberty sprawia, że wdrożenie nowej wersji jest proste

Pominięcie wydania nie wiąże się z odatkową pracą migracyjną

Tradycyjne użycie „fix pack”



Zastosowanie ciągłej integracji



Zastosuj wydania Continuous Integration i nigdy więcej nie stosuj ifix

Zero migracji

CI/CD Optimized

- Brak zmian w zachowaniu konfiguracji
- Brak zmian w zachowaniu funkcji runtime
- Brak usunięć funkcjonalności
- Pełne wydanie co 4 tygodnie

Bądź na bieżąco dzięki przebudowie
(bez konieczności zmiany aplikacji lub konfiguracji)

Pominięcie wydania nie wiąże się z
dodatkowymi pracami migracyjnymi

Bez wysiłku wyeliminuj dług
technologiczny i zachowaj
bezpieczeństwo



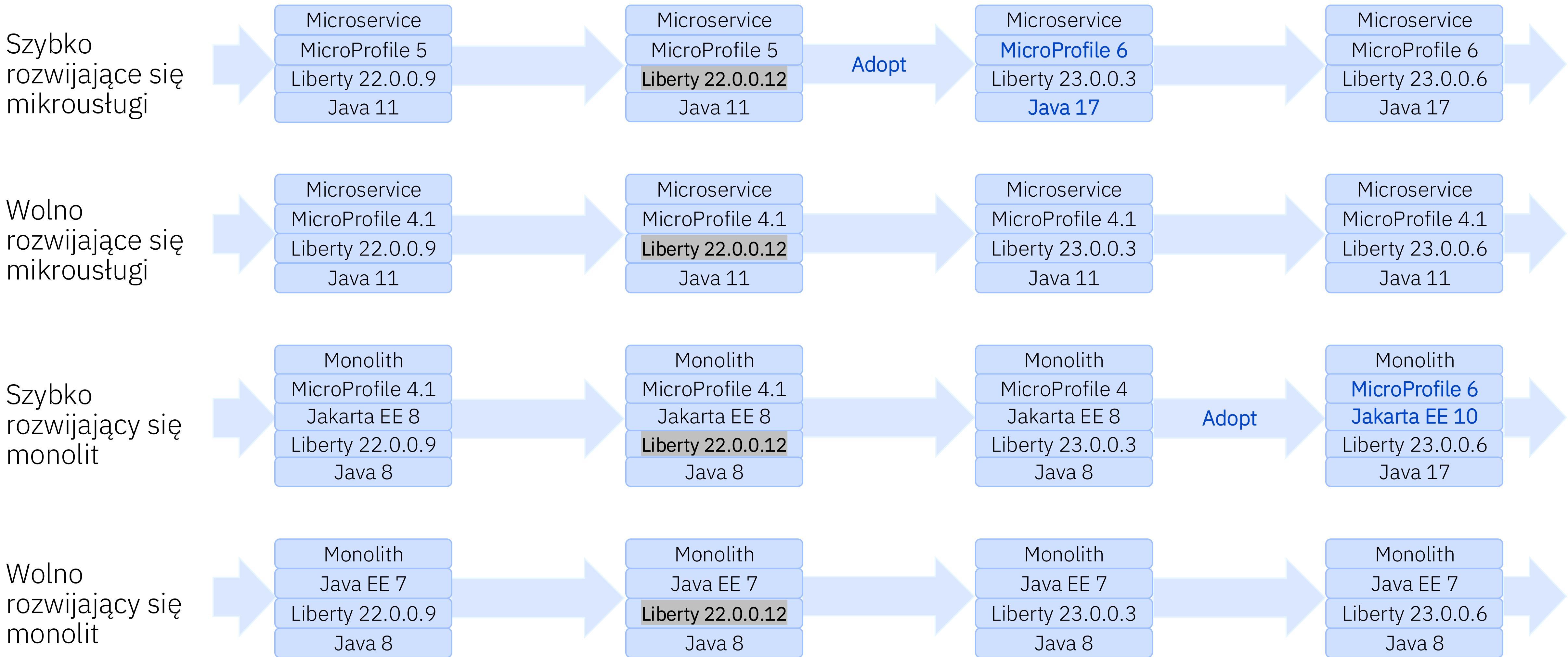
Liberty Zero Migracji

Co z innymi serwerami aplikacji??

- Inne serwery aplikacji mają poważne zmiany w każdej wersji.
 - Migracja z wersji na wersję może trwać nawet 2 lata i często przynosi ograniczone korzyści biznesowe.
-
- Modułowa architektura Liberty umożliwia obsługę starszych interfejsów API bez rozbudowywania wdrożeń.
 - Starsze interfejsy API są w pełni utrzymywane, zyskując najnowsze poprawki i ulepszenia wydajności
 - Bądź na bieżąco z Liberty i zmieniaj swój kod i konfigurację tylko wtedy, gdy chcesz zaadoptować nowe możliwości.

Specifications	Liberty	JBoss EAP 7.4	Tomcat 10.1.x	Quarkus
Jakarta EE 10	✓			MicroProfile subset
Jakarta EE 9/9.1	✓		WebProfile Subset	
Jakarta EE 8	✓	✓		
Java EE 8	✓	✓		
Java EE 7	✓			
Java EE 6*	✓			
MicroProfile 6.0	✓			
MicroProfile 5.0	✓			
MicroProfile 4.1	✓			✓
MicroProfile 4.0	✓			
MicroProfile 3.3	✓	✓		
MicroProfile 3.2	✓			
MicroProfile 3.0	✓			
MicroProfile 2.2	✓			
MicroProfile 2.1	✓			
MicroProfile 2.0	✓			
MicroProfile 1.4	✓			
MicroProfile 1.3	✓			
MicroProfile 1.2	✓			
MicroProfile 1.1	✓			
MicroProfile 1.0	✓			
Java SE	8, 11, 17, 21	8, 11, 17	11 or later	11, 17

Liberty Zero Migration – Adopcja na Twoich warunkach



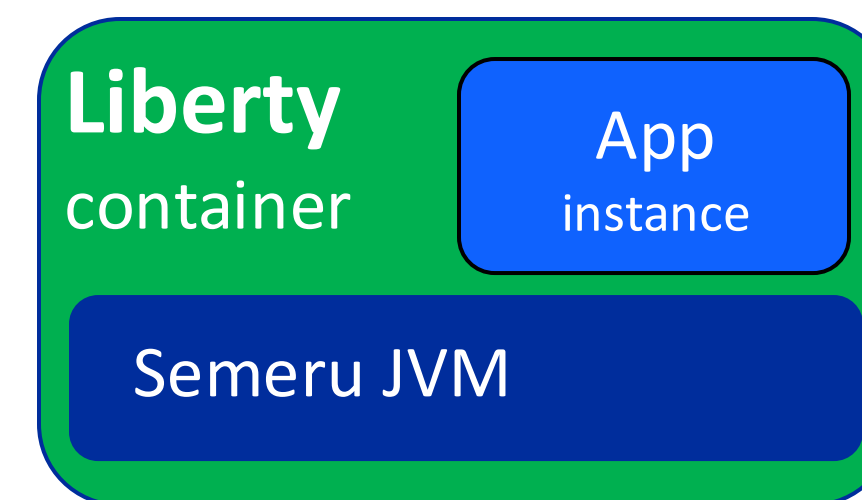
- Jakarta EE 10
- MicroProfile 6

5. Zoptymalizowane pod Kubernetes

Auto-tuning, szeroki wybór bazowych kontenerów

Pełne zastosowanie innowacji Java dla chmury

Optymalizacja kontenerów w chmurze na poziomie całego stosu Java



Prost dockerfile dla kontenerów aplikacji

Bezpieczny dostęp z IBM Container Registry (icr.io)

Zoptymalizowana
konfiguracja

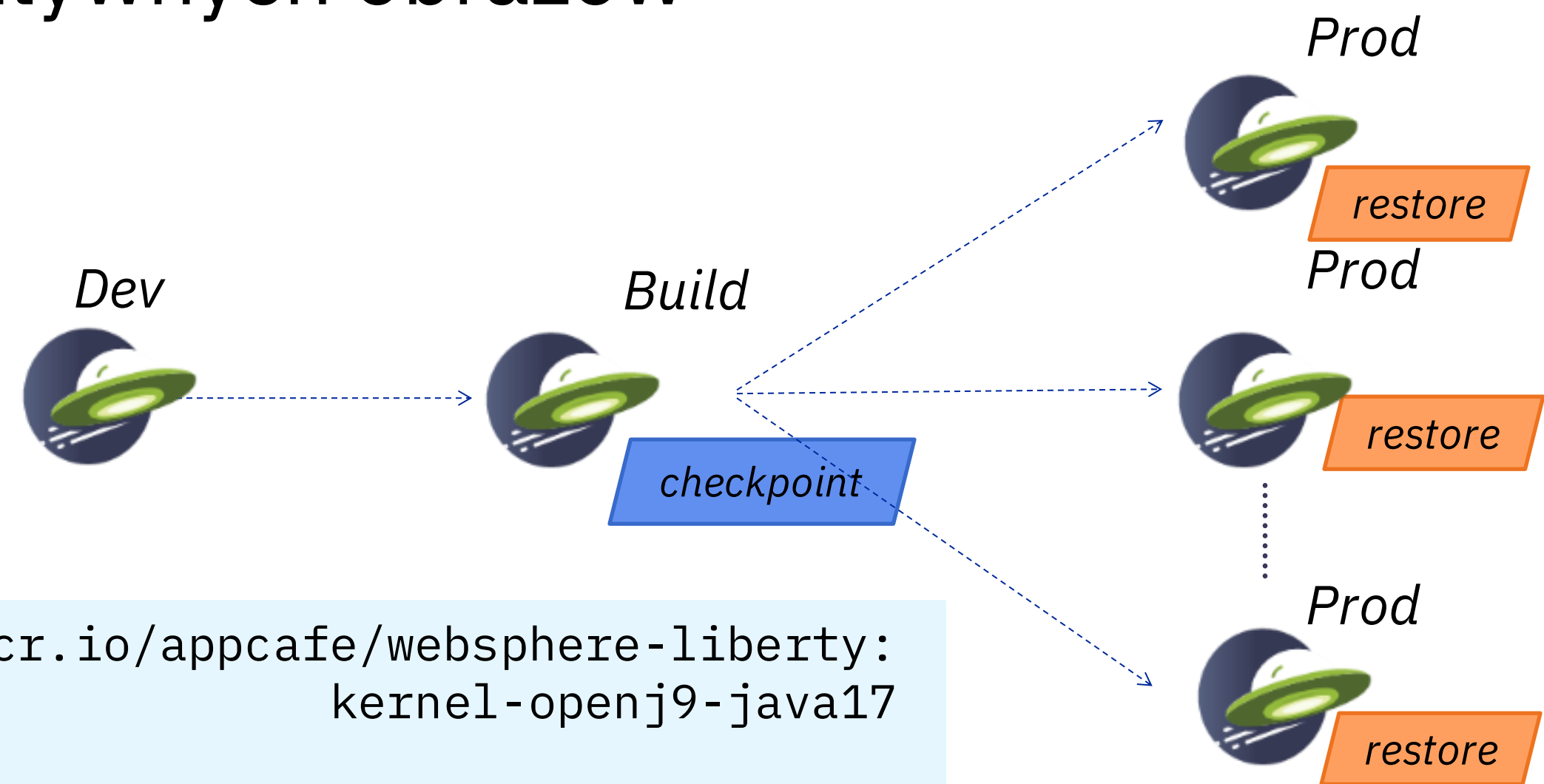
Zoptymalizowane
wykonanie

Containerfile

```
FROM icr.io/appcafe/open-liberty:kernel-slim-java17-openj9-ubi
COPY --chown=1001:0 /src/main/liberty/config/config
RUN features.sh
COPY --chown=1001:0 target/*.war /config/apps
RUN configure.sh
```


Wbudowany “InstantOn” Java

- Uruchamianie aplikacji w milisekundach
- Idealne rozwiązanie dla serverless
- Do 10-18 razy szybciej
- Ze wszystkimi zaletami JVM i bez ograniczeń natywnych obrazów



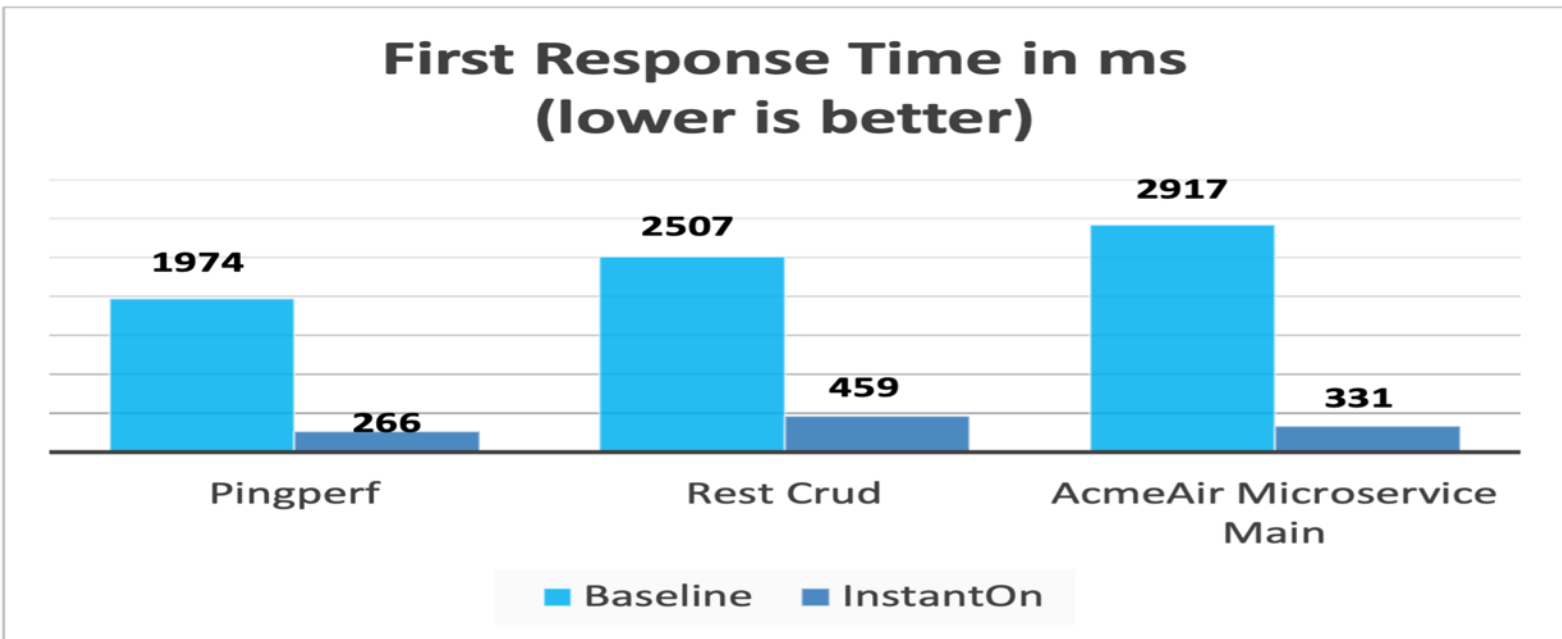
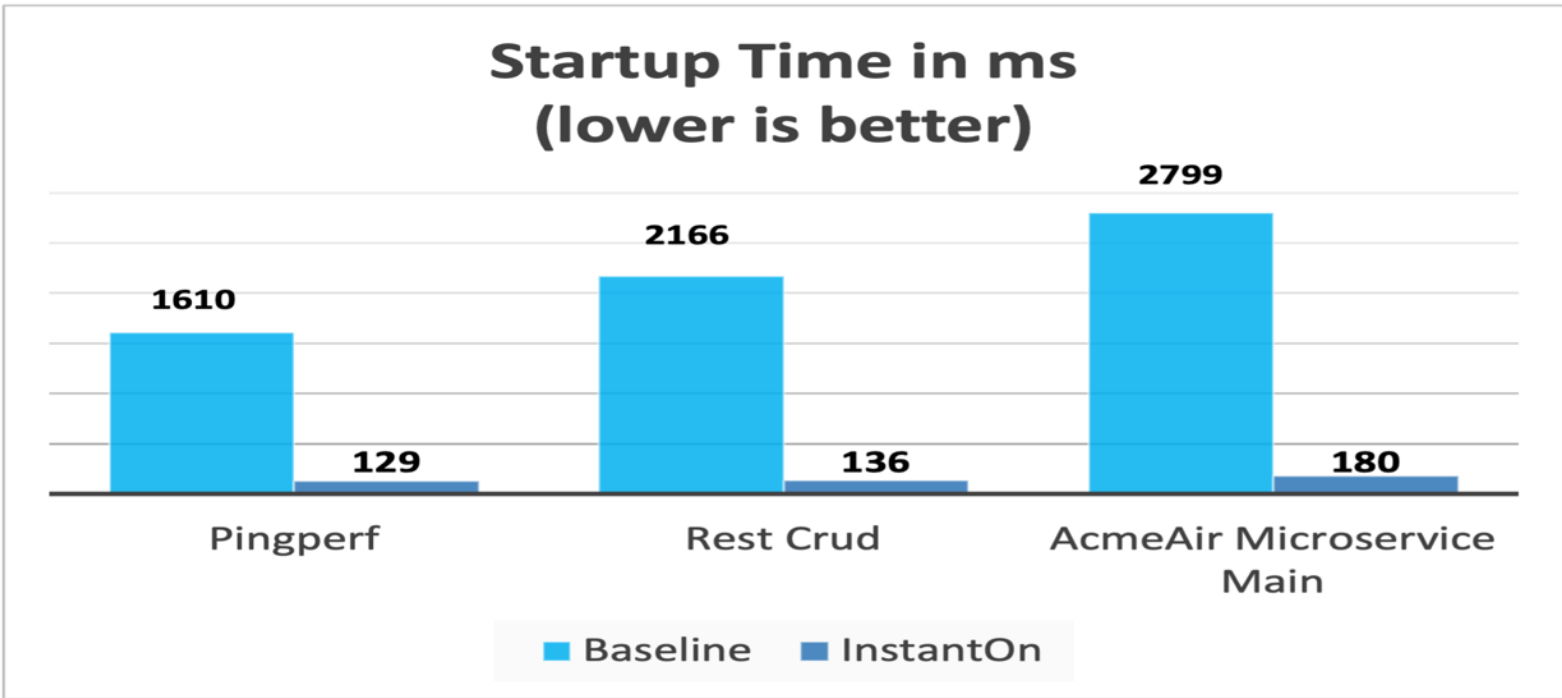
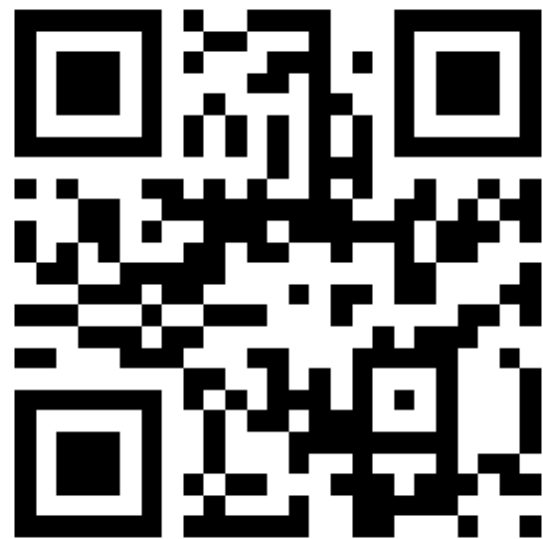
```
FROM icr.io/appcafe/websphere-liberty:
    kernel-openj9-java17

COPY src/main/liberty/config/*.xml
    /config/server.xml

RUN features.sh

COPY target/*.war /config/apps/

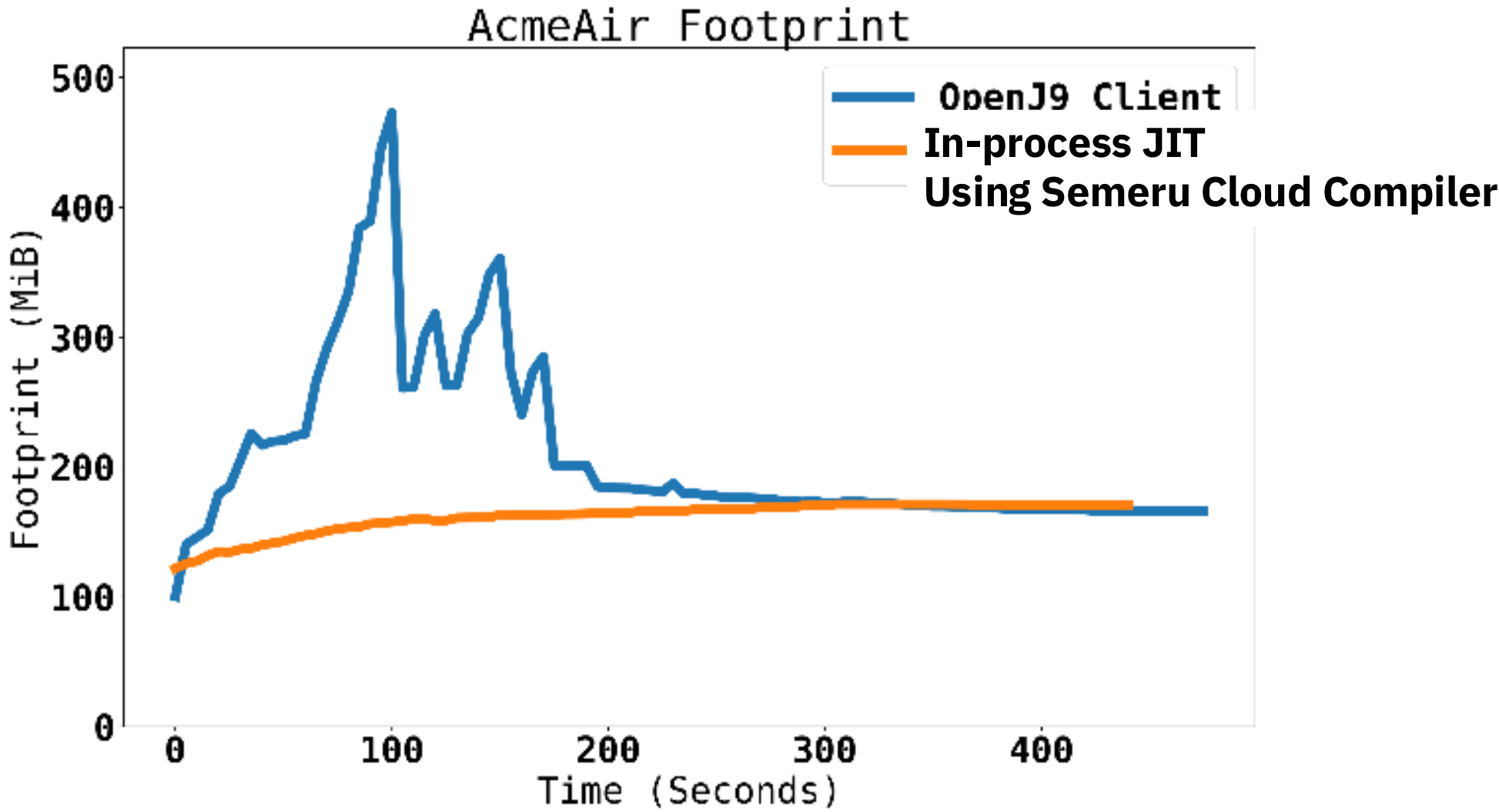
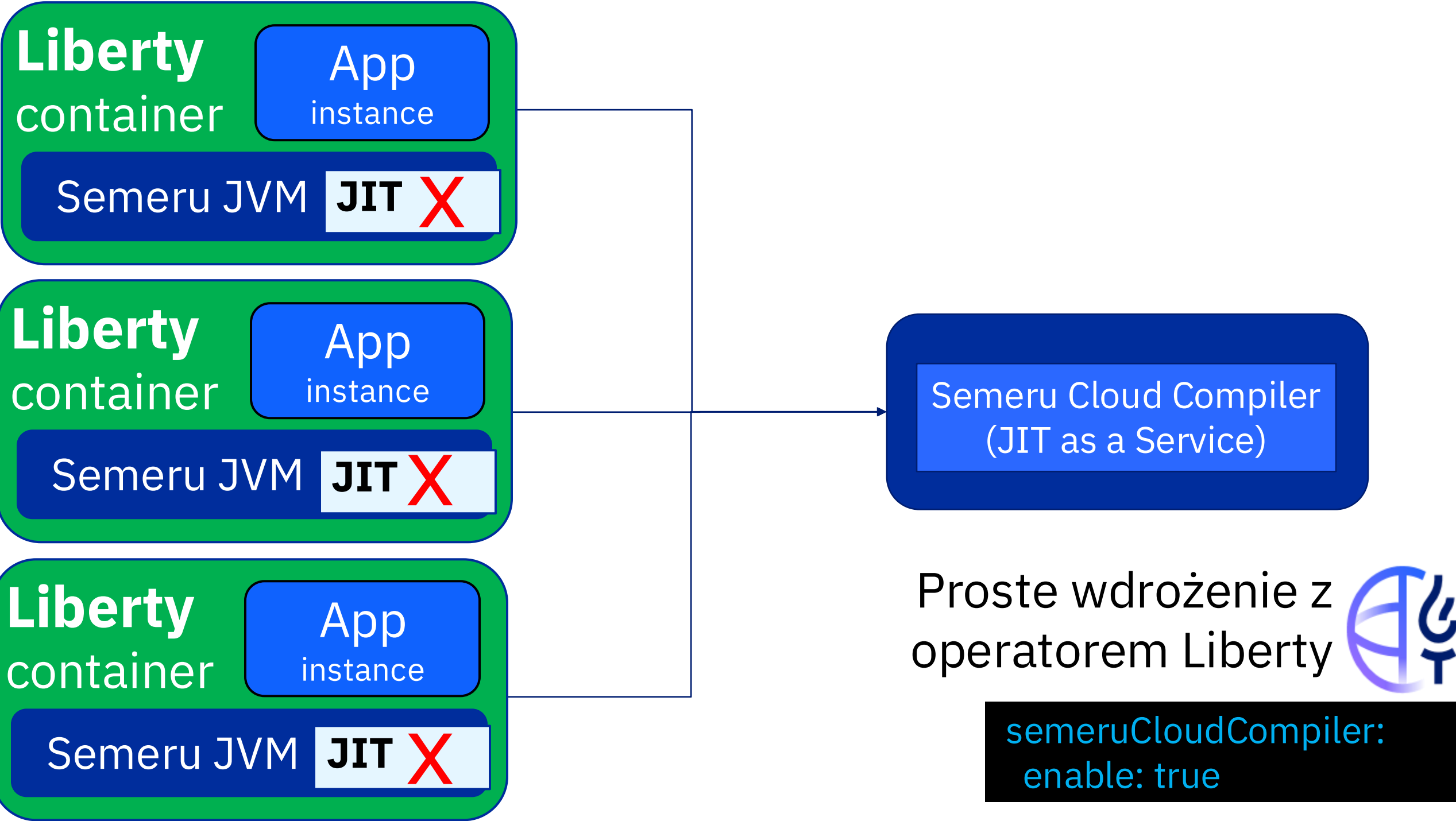
RUN configure.sh
RUN checkpoint.sh afterAppStart
```



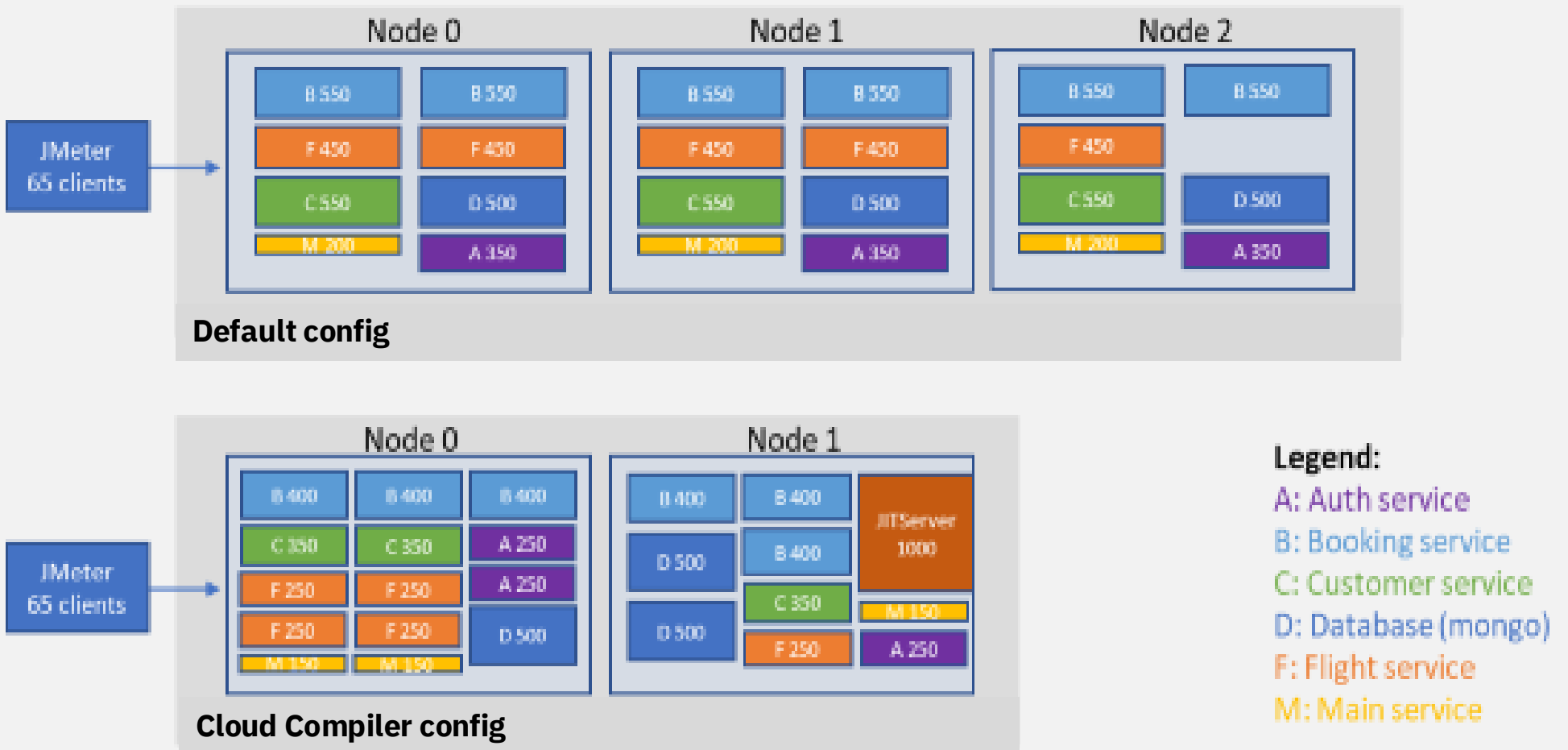
Characteristics	Semeru InstantOn	Semeru JVM	Graal Native
Full Java support	Yes	Yes	No
‘Instant on’	Yes	No	Yes
High throughput	Yes	Yes	No
Low memory (under load)	Yes	Yes	No
Dev-prod parity	Yes	Yes	No

Zwiększ wydajność operacyjną dzięki Cloud Optimizer

- Odciąża pamięć i procesor intensywną kompilacją JIT do oddzielnego serwera
- Większa gęstość instancji aplikacji przy tej samej przepustowości -> niższy koszt



Cloud Compiler większa gęstość → mniej węzłów → mniejszy koszt



Prostota wdrażania i zarządzania - operator Liberty

Rozwiązanie problemu luki w umiejętnościach związanych z Kubernetes

- Redukcja konfiguracji nawet o 80%
- Automatyzacja typowych zadań: wdrażanie, skalowanie, aktualizacja, gromadzenie zrzutów
- Gotowe do użycia funkcje bezpieczeństwa*
- Obsługa niestandardowych zasobów aplikacji Liberty, które rozszerzają Kubernetes
- Izolacja od złożoności Kubernetes



Liberty custom resource
Processed by **Liberty operator**

```
apiVersion: liberty.websphere.ibm.com/v1
kind: WebSphereLibertyApplication
metadata:
  name: liberty-cloud-demo
spec:
  license:
    accept: false
    edition: IBM WebSphere Application Server
    productEntitlementSource: Standalone
    metric: Processor Value Unit (PVU)
  replicas: 3
  applicationImage: liberty-demo:1.0
  pullPolicy: Always
  expose: true
  semeruCloudCompiler:
    enable: true
```

<https://openliberty.io/docs/latest/open-liberty-operator.html>

* Przykłady: Integracja zarządzania certyfikatami z OCP, delegowanie SSO

Utrzymanie

Właściwe informacje we właściwym miejscu i we właściwym czasie

Monitorowanie i diagnostyka

Monitorowanie wspierane przez runtime, gotowe do integracji z zewnętrznymi systemami

Operacje Dnia-2

Tracing i zrzuty (dumps) umożliwiające poprzez Liberty Operator



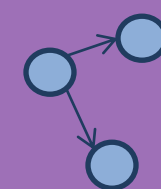
Logging

- JSON logging (logs, trace, ffdc, access logs, audit logs)
- Integration with java logging API (JUL)
- LogRecordContext (add custom fields to log records)
- LogstashCollector feature, Kibana Dashboards
- Works with common log aggregators (Splunk, Humio, LogDNA, Elastic Stack, etc)



Metrics

- Built-in JVM and Liberty metrics
- App metrics using MicroProfile Metrics aggregation and dashboarding with Prometheus and Grafana



Tracing

- Distributed tracing using MicroProfile Telemetry
- Built-in Jakarta REST instrumentation
- App instrumentation using MicroProfile Telemetry
- Zipkin, Jaeger to aggregate/visualize
- OpenTelemetry under development

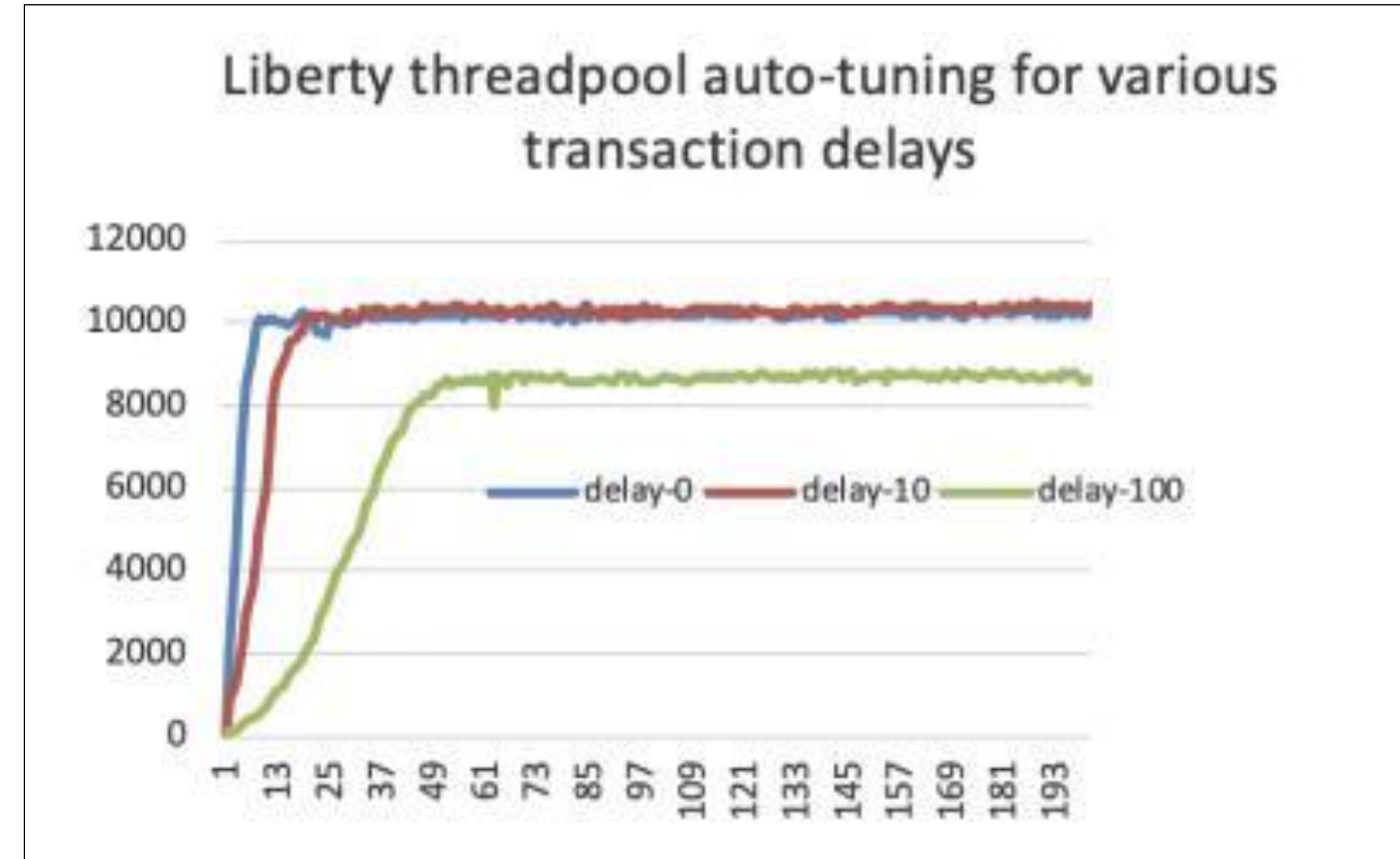


Health

- Kube Health endpoint using MicroProfile
- Startup, Liveness, Readiness for different lifecycle states
- App health checks using MicroProfile

Auto-tuning runtime

Liberty automatycznie dostraja się, aby zoptymalizować przepustowość dla dostępnych zasobów



“You don't have to tune thread pools. Liberty does an outstanding job”
– Shawn Hisaw, WAS Technology Owner at a large health provider

6. Zoptymalizowany dla deweloperów

Interfejsy API neutralne dla dostawców, tryb deweloperski, obsługa kontenerów, CI/CD

Standardowe Interfejsy API

Interfejsy mikrouslug i aplikacji oparte na standardach (Microprofile, Java/Jakarta EE) wolne od ograniczeń związanych z dostawcami

<https://microprofile.io/compatible/>
<https://jakarta.ee/compatibility/>

- Buduj nowe otwarte mikrouslugi za pomocą MicroProfile, wykorzystując istniejące umiejętności i zasoby Java EE/Jakarta EE
- Modernizuj istniejące aplikacje Java EE do środowiska natywnego dla chmury za pomocą Jakarta EE i MicroProfile

*Optimizing Enterprise Java
for a Microservices Architecture*



2022: First MicroProfile 5.0 and 6.0
Compatible runtimes



Jakarta EE 10 released Sep 2022
Compatible implementation on day of release

Java EE

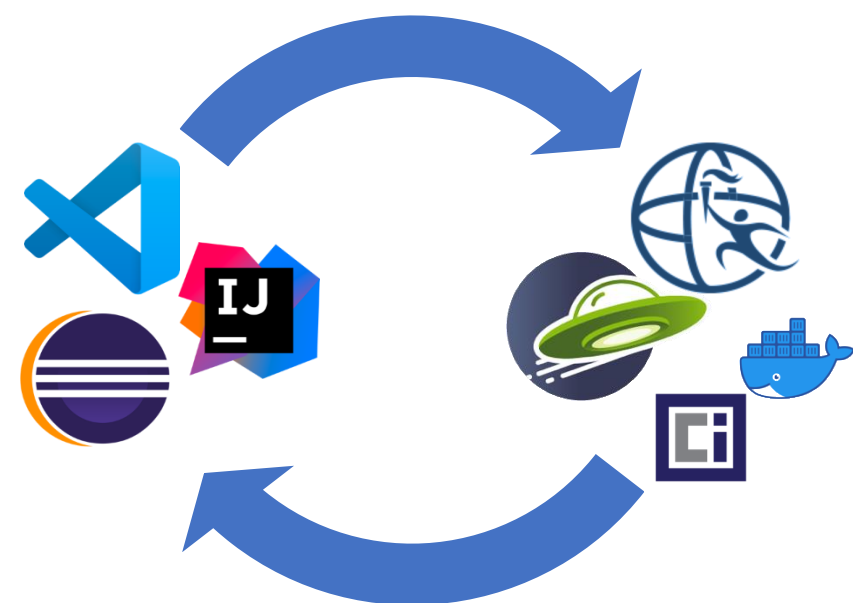
*Build modern portable enterprise apps
Protect your investments in Java EE*



Liberty Blogs: [Jakarta EE 10 & MicroProfile 6 support in Open Liberty 23.0.0.3](#)

Dev Mode

- Bez przebudowy
- Brak ponownego wdrożenia
- Brak instalacji
- Bez restartu
- Tylko kod!
- Również w kontenerach



Wsparcie DevOps

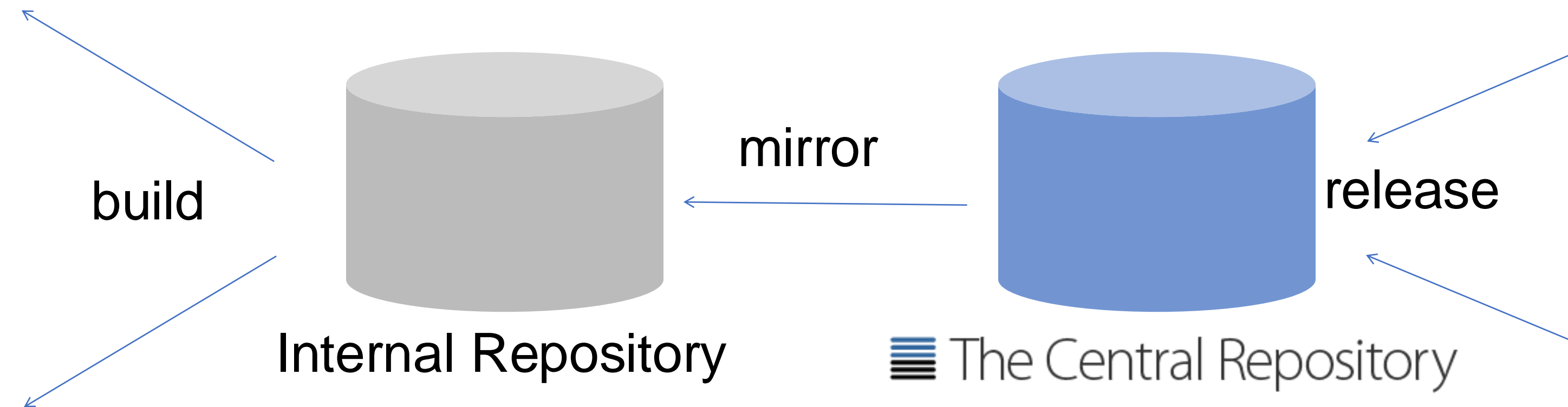


maven

```
<plugin>
  <groupId>io.openliberty.tools</groupId>
  <artifactId>liberty-maven-plugin</artifactId>
  <version>3.2</version>
</plugin>
```


gradle

```
dependencies {
  classpath 'io.openliberty.tools:liberty-gradle-plugin:3.0'
}
```



Poradniki Open Liberty

- 55 poradników
 - MicroProfile & Jakarta EE
 - Open Shift, Docker, Kubernetes Istio

[Get Started](#)[Guides](#)[Docs](#)[Support](#)[Blog](#)

Guides

The quickest way to learn all things Open Liberty, and beyond!

Filter guides

X

DEVELOP (37 guides)

Getting started

RESTful service

Reactive service

Configuration

Fault tolerance

Observability

Security

Persistence

Client side

BUILD AND TEST (9 guides)

Build

Test

Containerize

DEPLOY (9 guides)

Kubernetes

Cloud deployment

Developing your cloud-native application

Getting started

RESTful service

Getting started with Open Liberty

Learn how to develop a Java application on Open Liberty with Maven and Docker.

🕒 25 minutes

Injecting dependencies into microservices

Learn how to use Contexts and Dependency Injection (CDI) to manage and inject dependencies into microservices.

🕒 15 minutes

Creating a RESTful web service

Learn how to create a REST service with JAX-RS, JSON-B, and Open Liberty.

🕒 30 minutes

Consuming RESTful services with template interfaces

Learn how to use MicroProfile Rest Client to invoke RESTful services over HTTP in a type-safe way.

🕒 20 minutes

Consuming a RESTful web service

Explore how to access a simple RESTful web service and consume its resources in Java using JSON-B and JSON-P.

🕒 25 minutes

Documenting RESTful APIs

Explore how to document and filter RESTful APIs from code or static files by using MicroProfile OpenAPI.

🕒 20 minutes

Creating a hypermedia-driven RESTful web service

Learn how to use Hypermedia As The Engine Of Application State (HATEOAS) to drive your RESTful web service

🕒 30 minutes

Consuming RESTful services asynchronously with template interfaces

Learn how to use MicroProfile Rest Client to invoke RESTful microservices asynchronously over HTTP.

🕒 15 minutes

