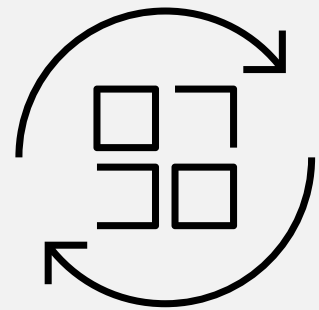


Nowoczesne aplikacje wymagają nowoczesnych narzędzi - w jaki sposób wspieramy modernizację aplikacji z Cloud Pak for Applications

Mikołaj Jaworski
Partner Technical Specialist @ IBM

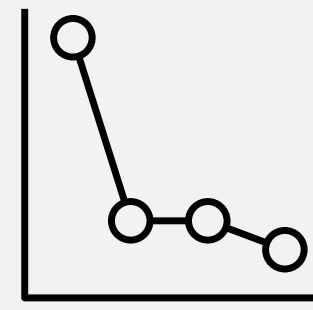


Co napędza modernizację?



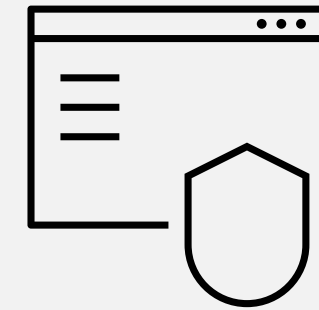
Speed-to-market

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



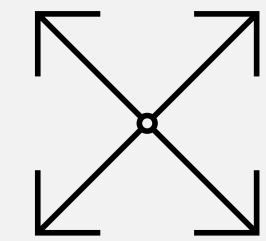
Redukcja kosztów

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 technicznego.



Dostęp do innowacji w oprogramowaniu

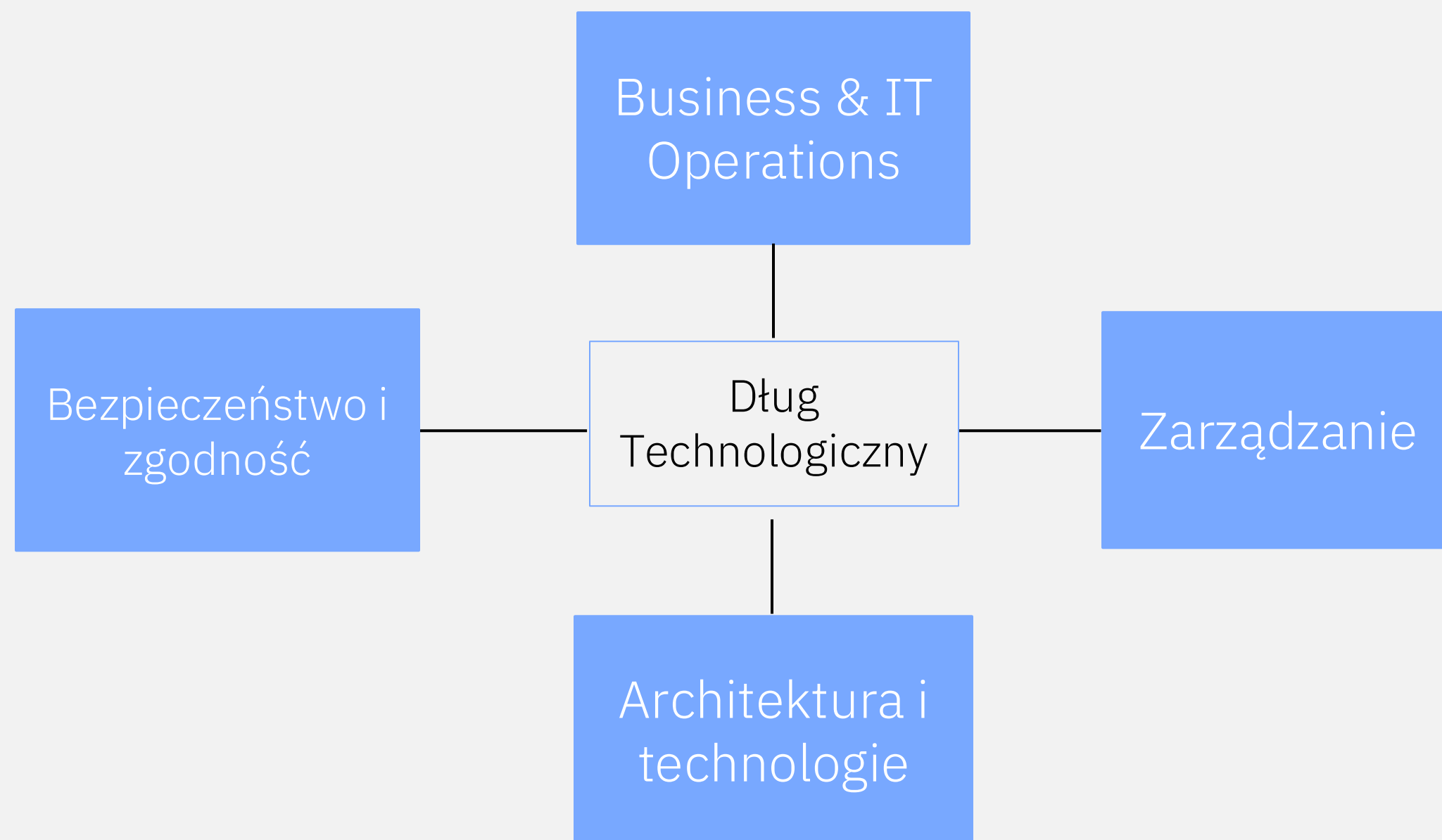
Wykorzystanie nowych technologii w celu przyspieszenia wyników biznesowych.

Dług technologiczny jest główną przeszkodą w osiągnięciu celów biznesowych

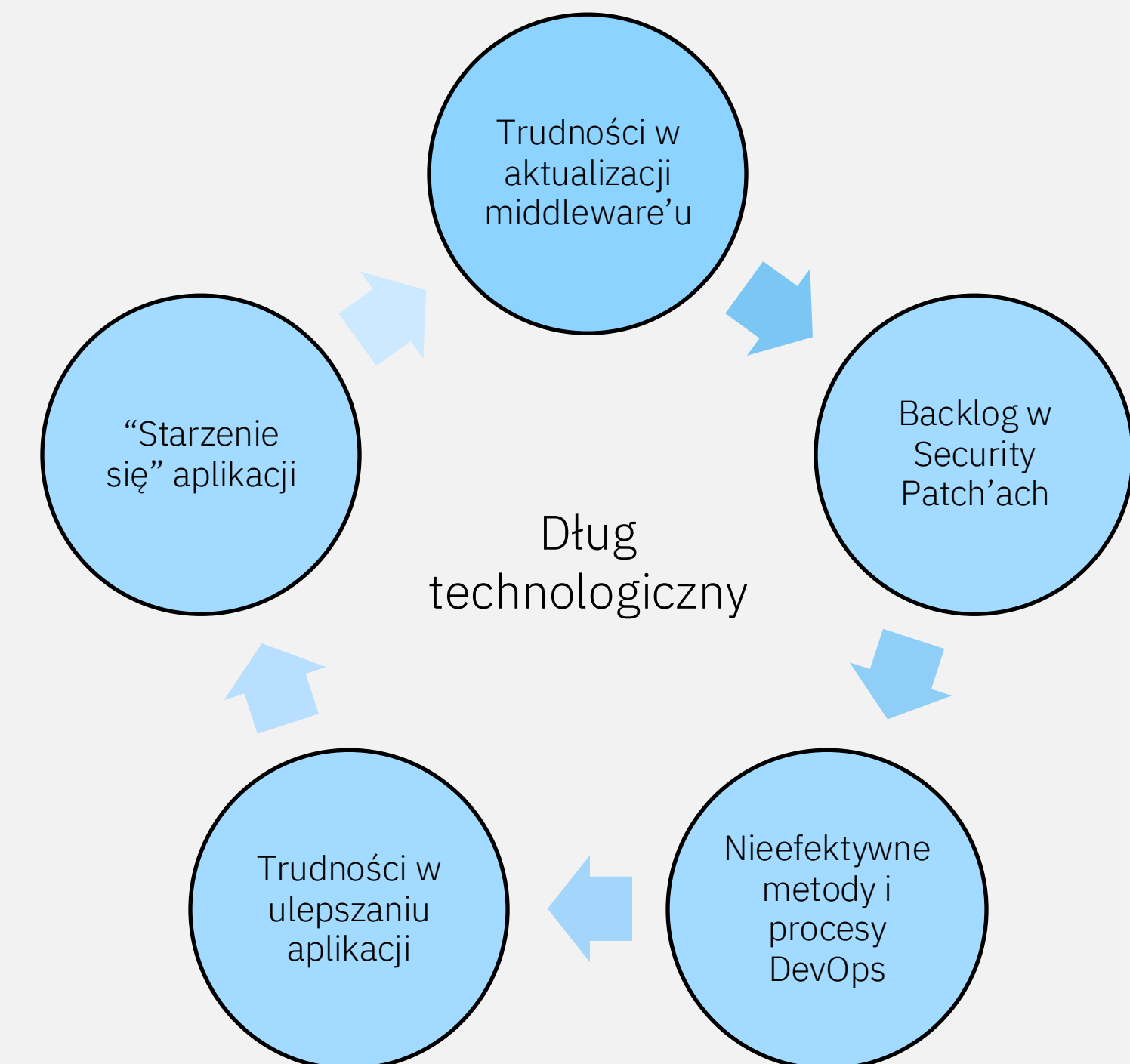
Co to jest?

„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

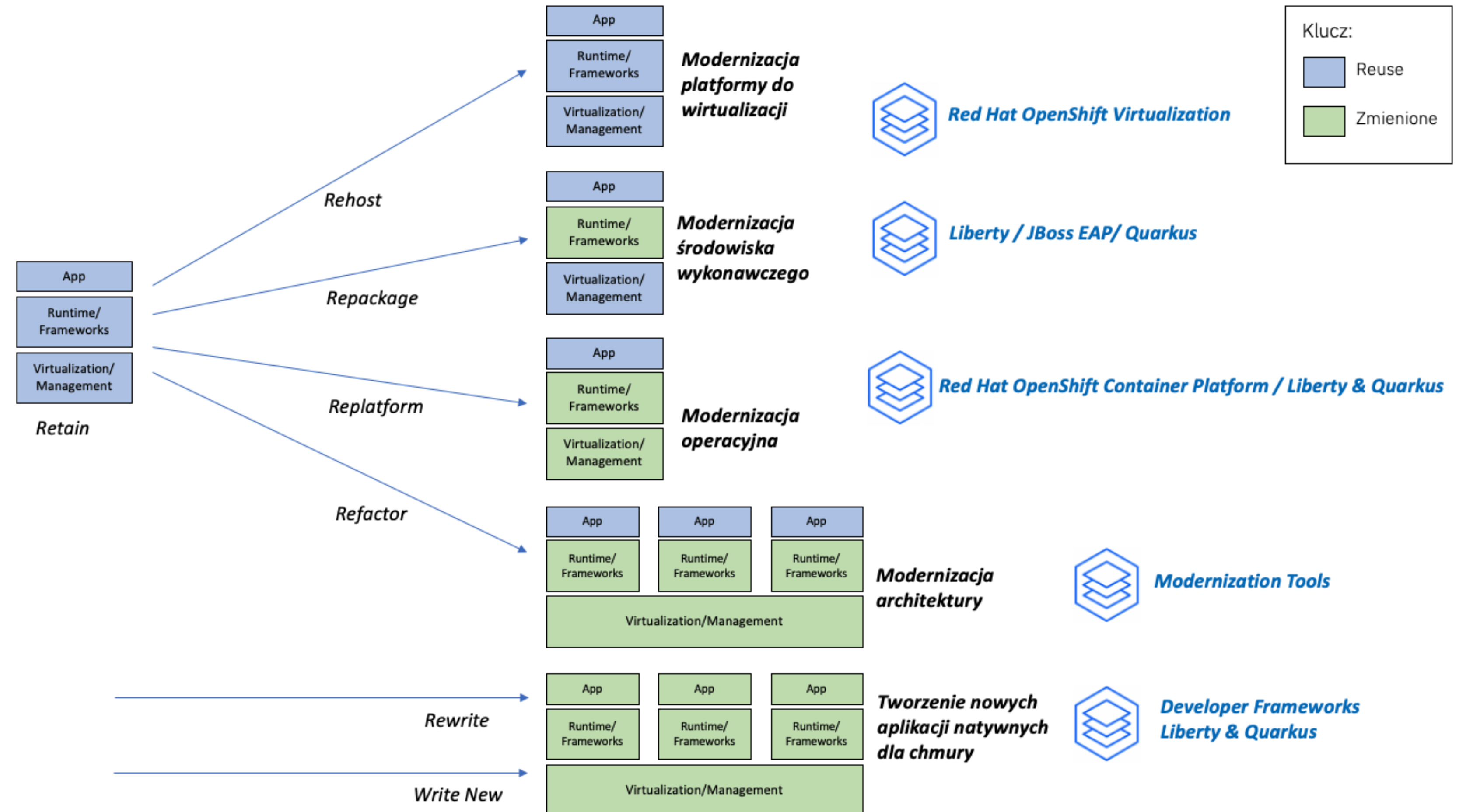


Podejście do modernizacji aplikacji z CP4Apps

Czynniki wpływające na modernizację:

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

Nie ma uniwersalnej odpowiedzi

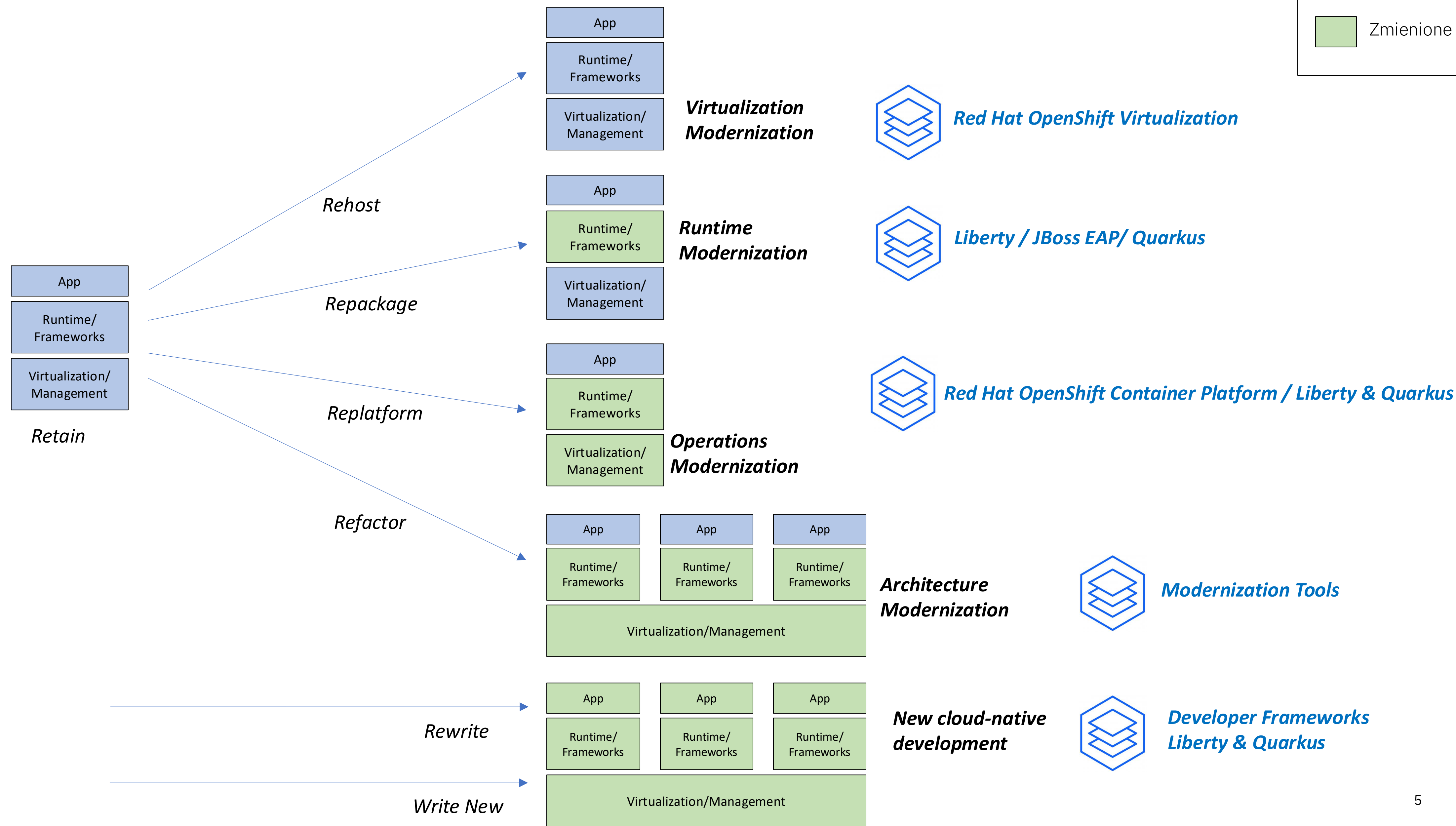


Podejście do modernizacji aplikacji z CP4Apps

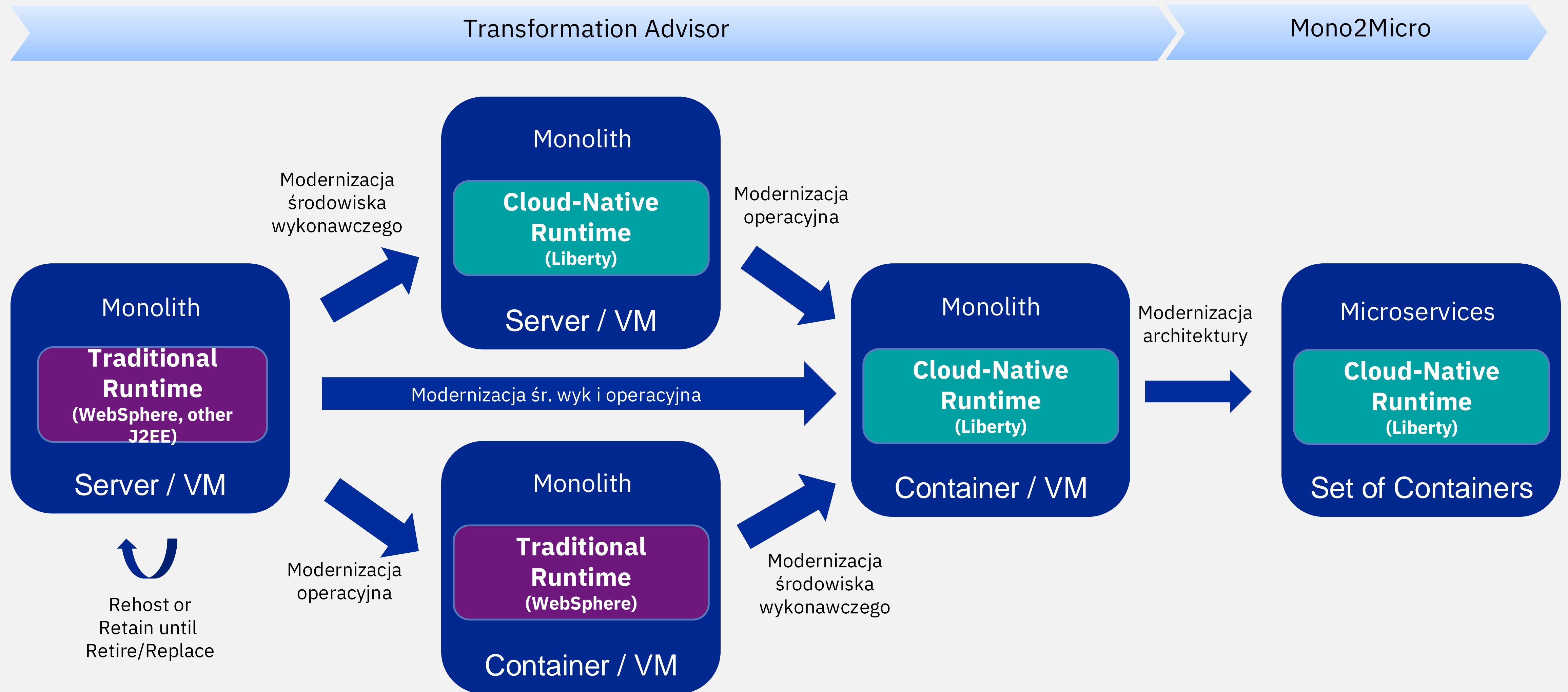
Klucz:

Reuse

Zmienione

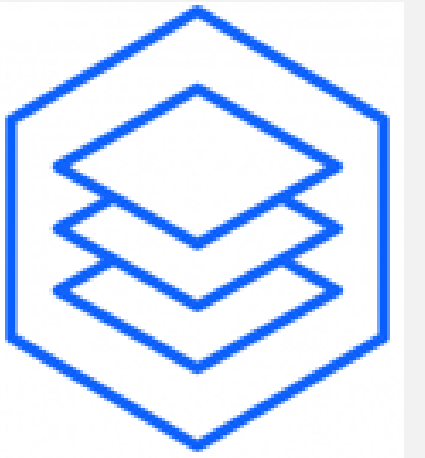


Jaki jest plan na modernizację?



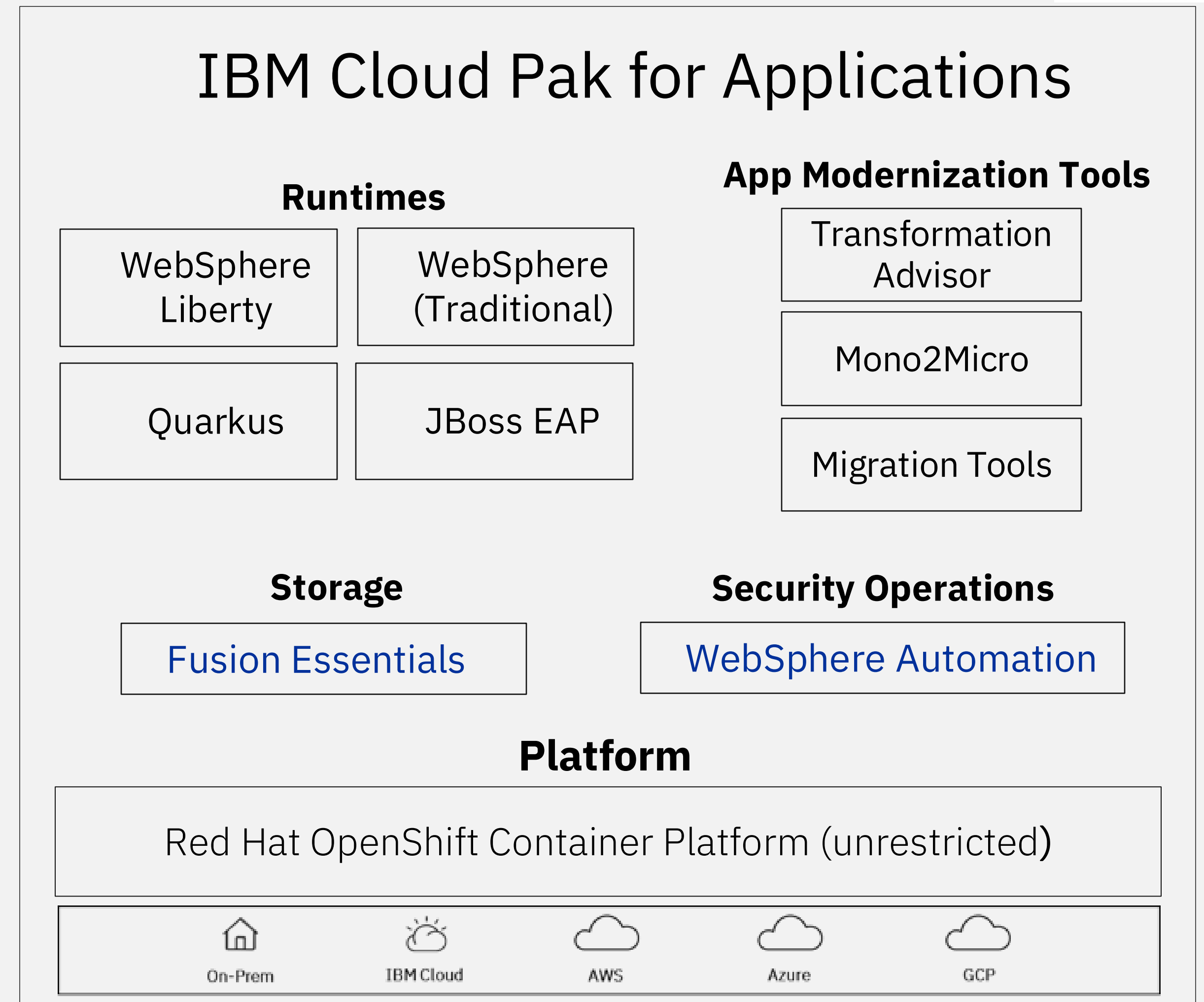
Czym jest IBM Cloud Pak for Applications?

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

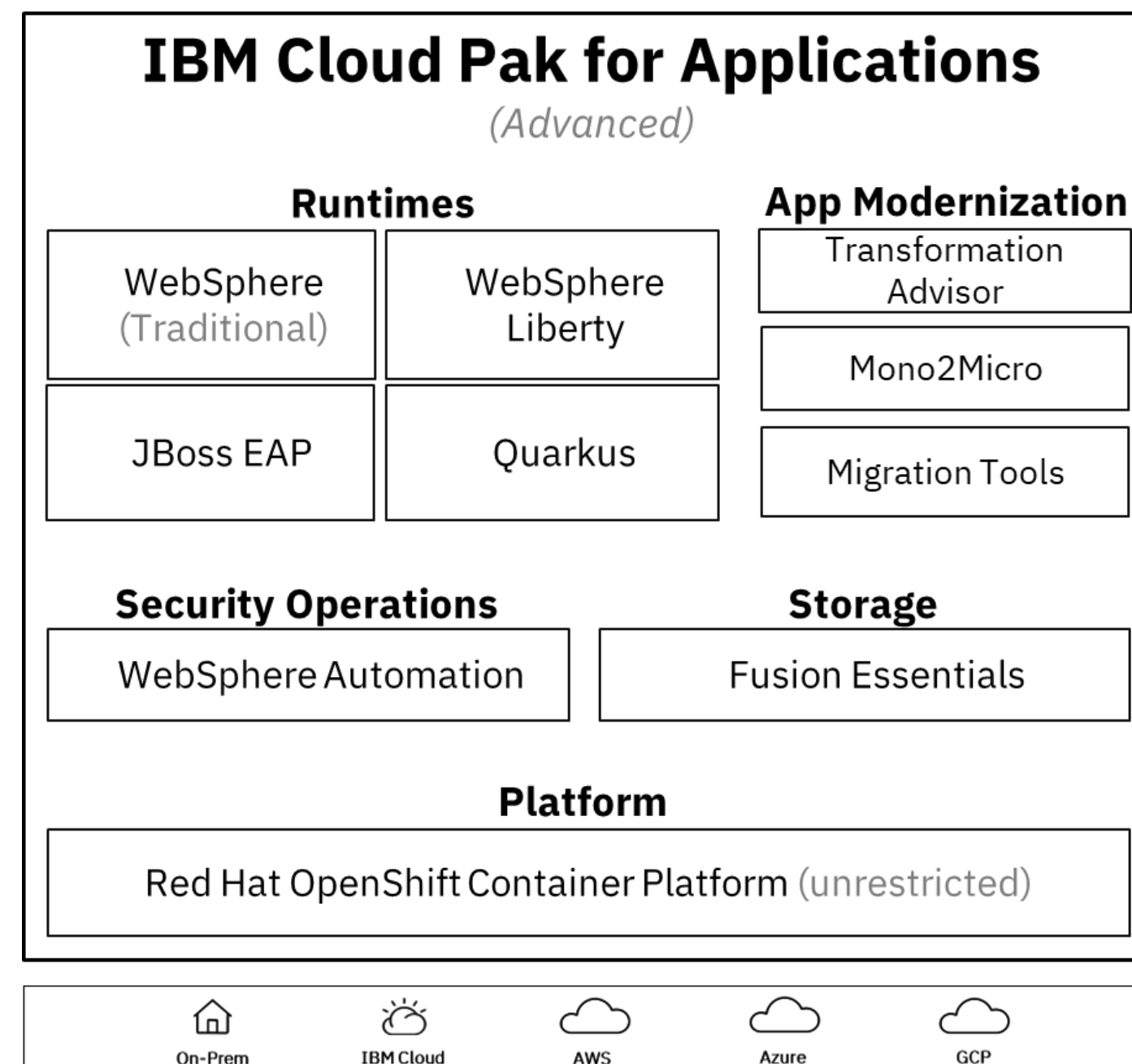
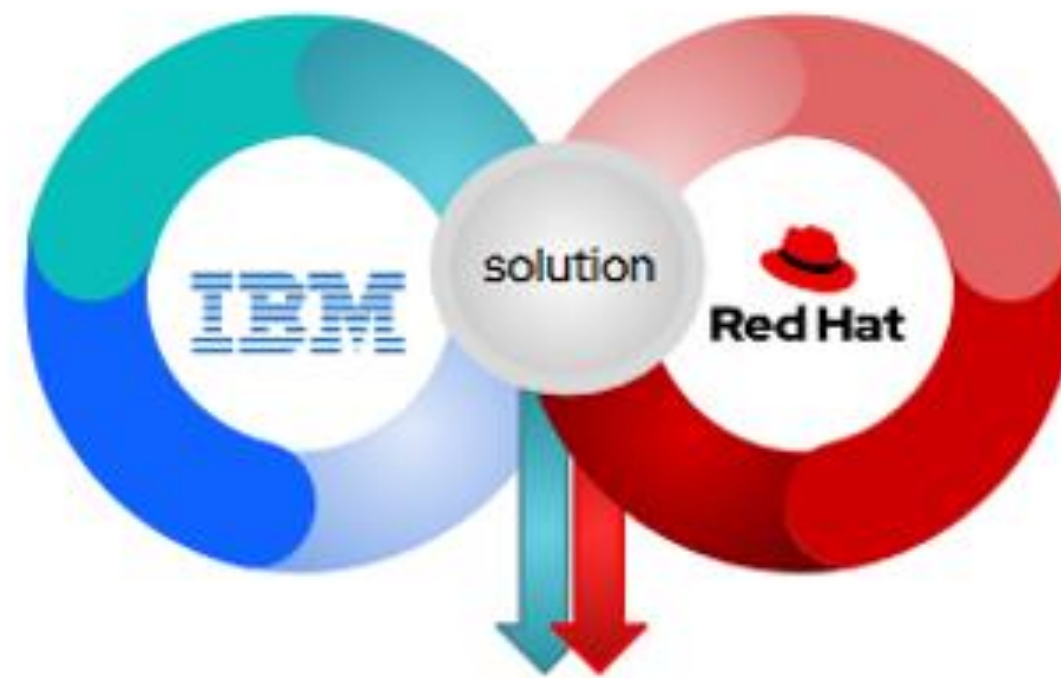
- Mono2Micro
- Transformation Advisor
- WebSphere Application Server Migration Toolkit

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

Porozmawiajmy o WebSphere Liberty

IBM WebSphere Liberty to środowisko wykonawcze nowej generacji, które przyspiesza uruchamianie aplikacji natywnych dla chmury.



Wspólne funkcje Open Liberty i WebSphere Liberty są identyczne, wydawane razem i wspierane w ramach oferty WebSphere.

Open Liberty

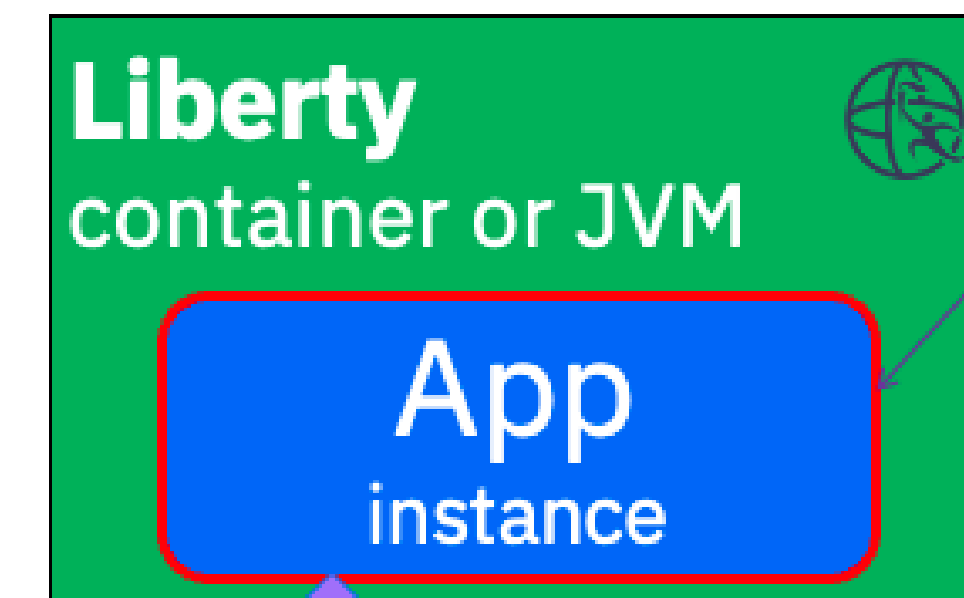
Zaprojektowany z myślą zarówno o programistach, jak i właścicielach aplikacji.

- Liberty dostarcza najnowsze interfejsy API Java i integruje się z najpopularniejszymi narzędziami deweloperskimi i kompilacyjnymi.
- 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/>

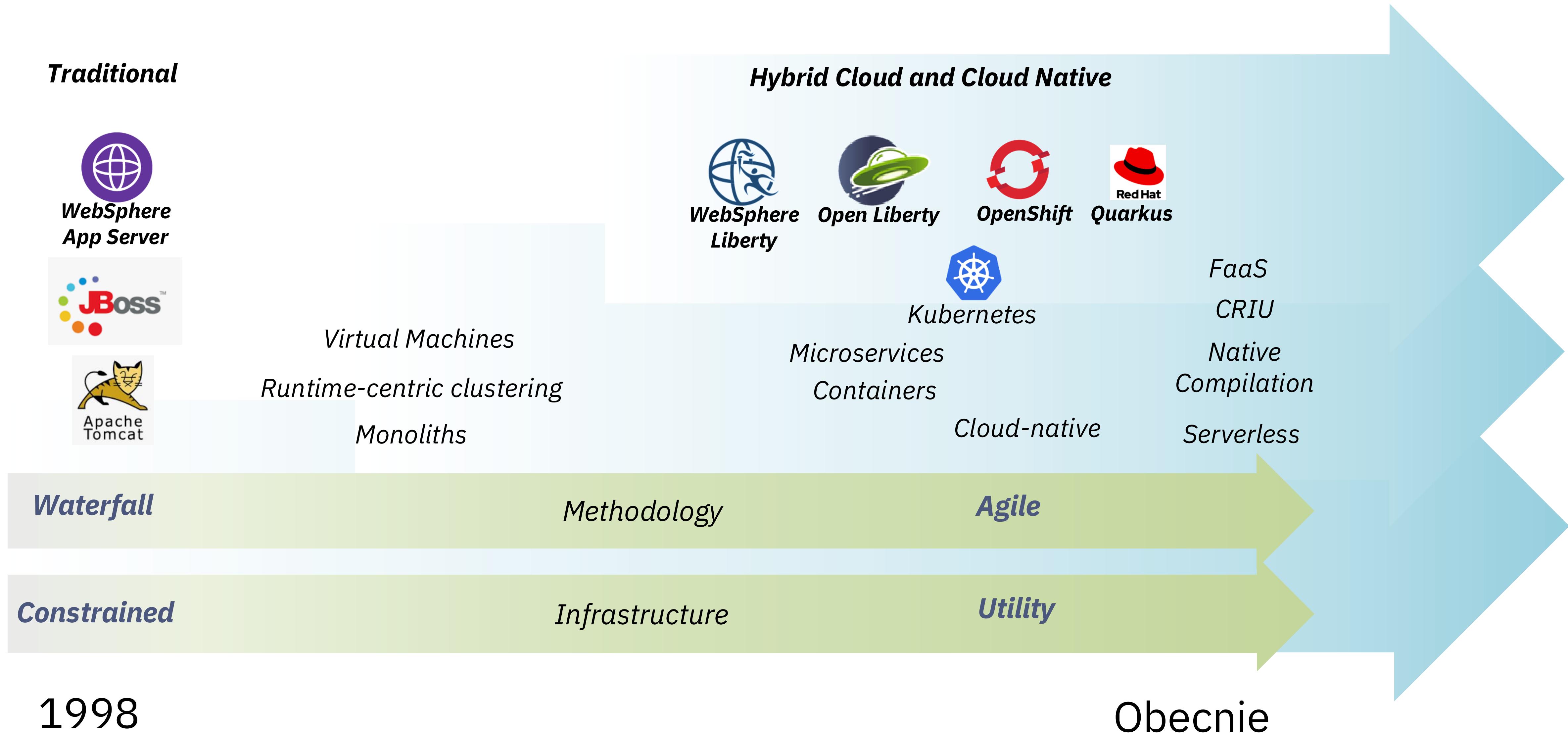


**MicroProfile and
Jakarta EE APIs for
Applications**

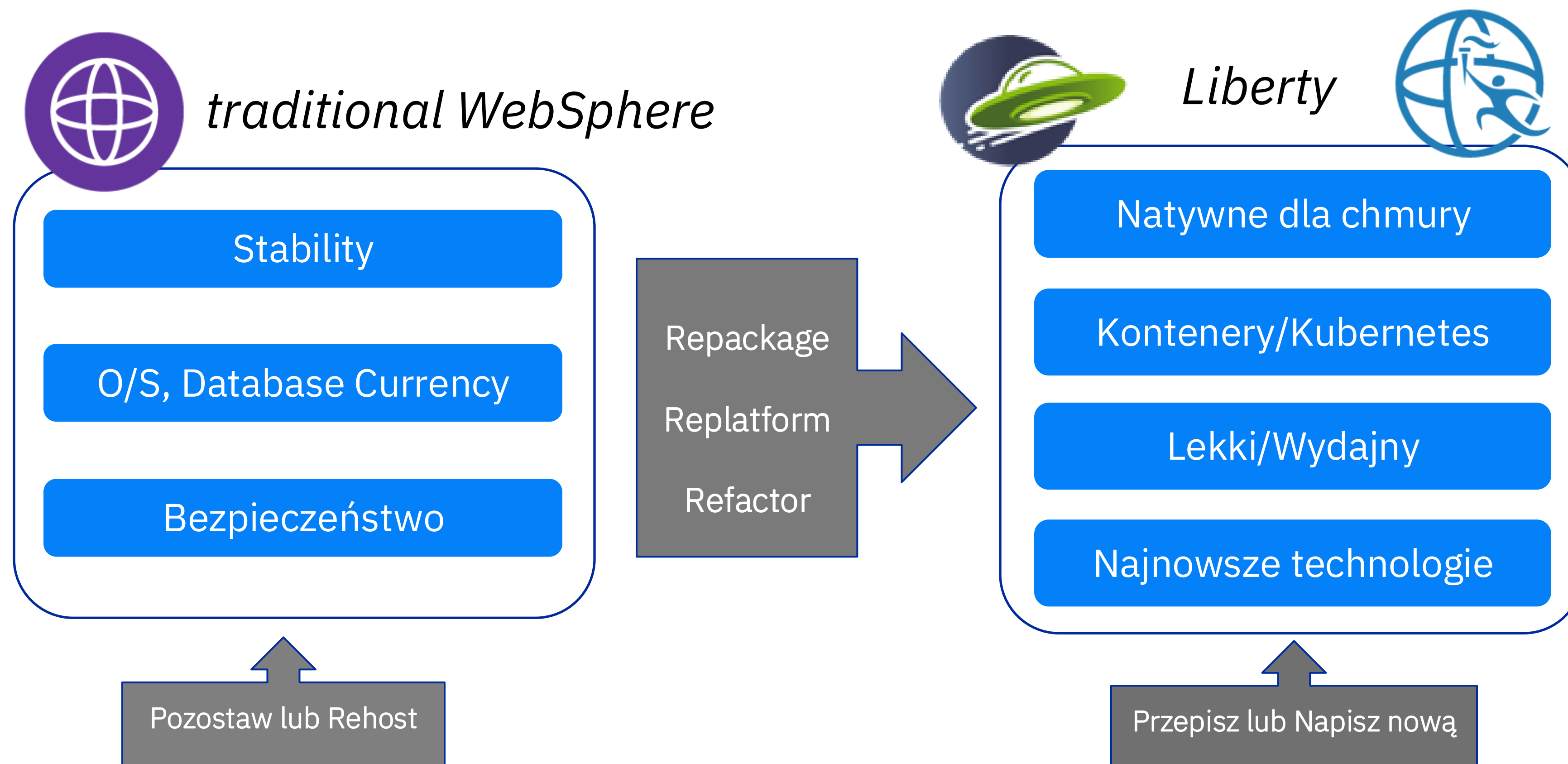
**Liberty Tools With All
Popular Developer
Environments (IDEs)**

Java jest jednym z najpopularniejszych języków programowania aplikacji

- Obsługuje architekturę monolityczną, makroserwisy i mikroserwisy ...

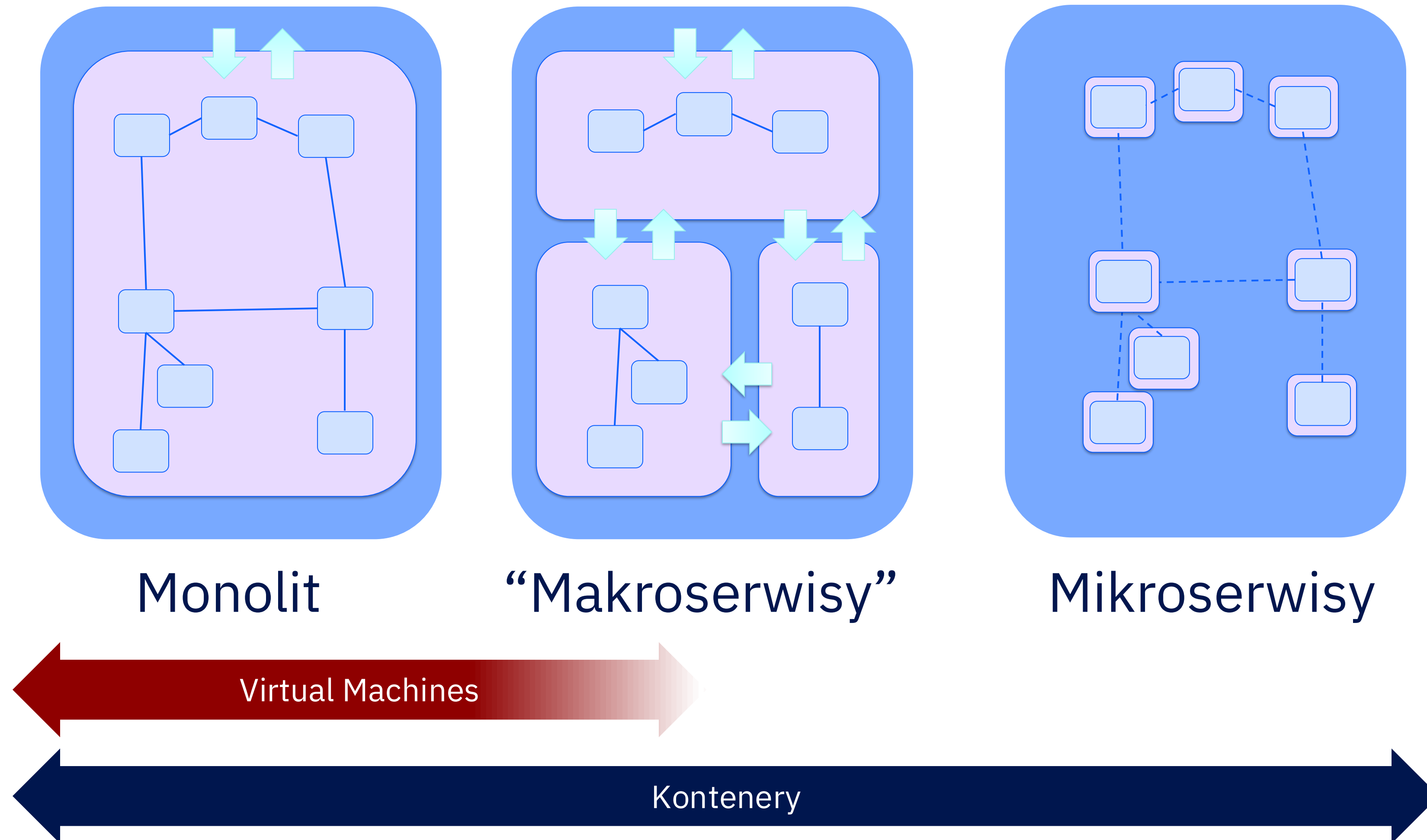


Wsparcie obecnych i przyszłych potrzeb w zakresie środowiska wykonawczego poprzez modernizację z WebSphere do Liberty



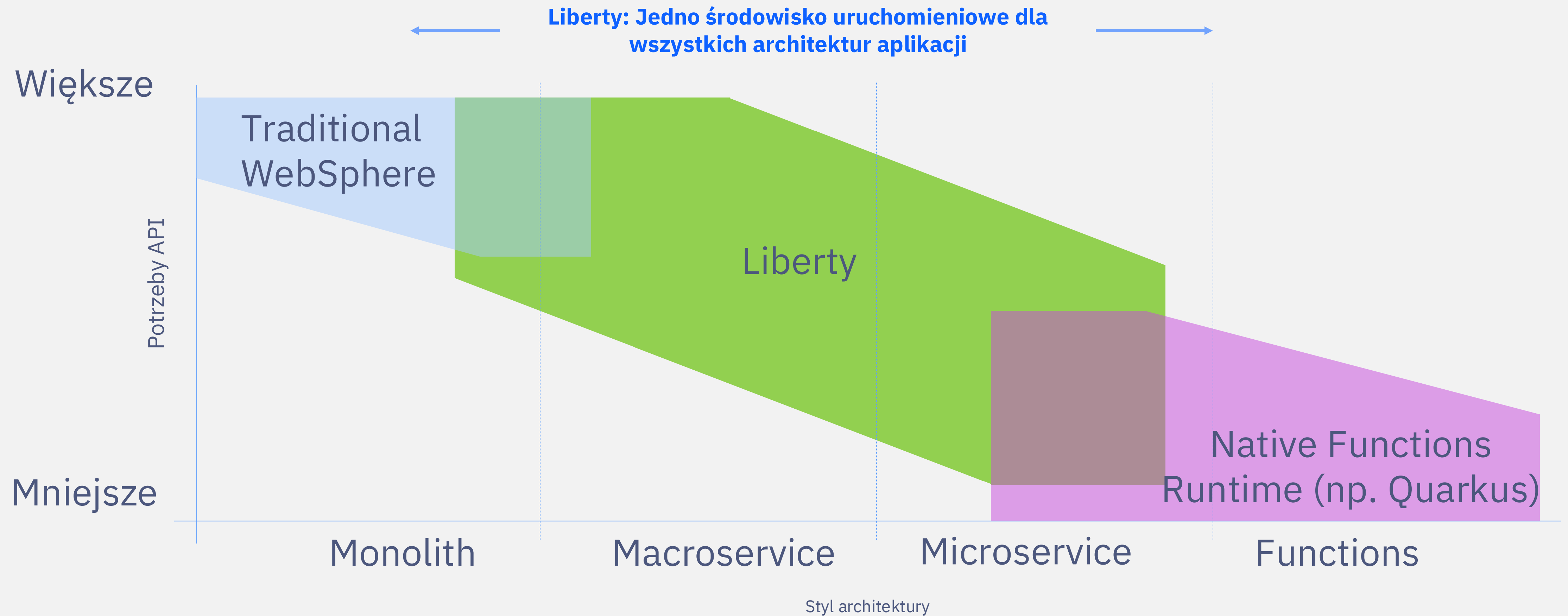
Liberty to środowisko uruchomieniowe IBM dla modernizacji Enterprise Java i nowych środowisk wykonawczych.

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



Odpowiednie środowisko wykonawcze Java

Potrzeby API dla różnych stylów architektury



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

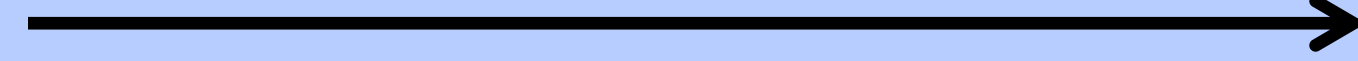
Poprawa bezpieczeństwa i niższe koszty administracyjne

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

1. Runtime taki jak potrzebujesz

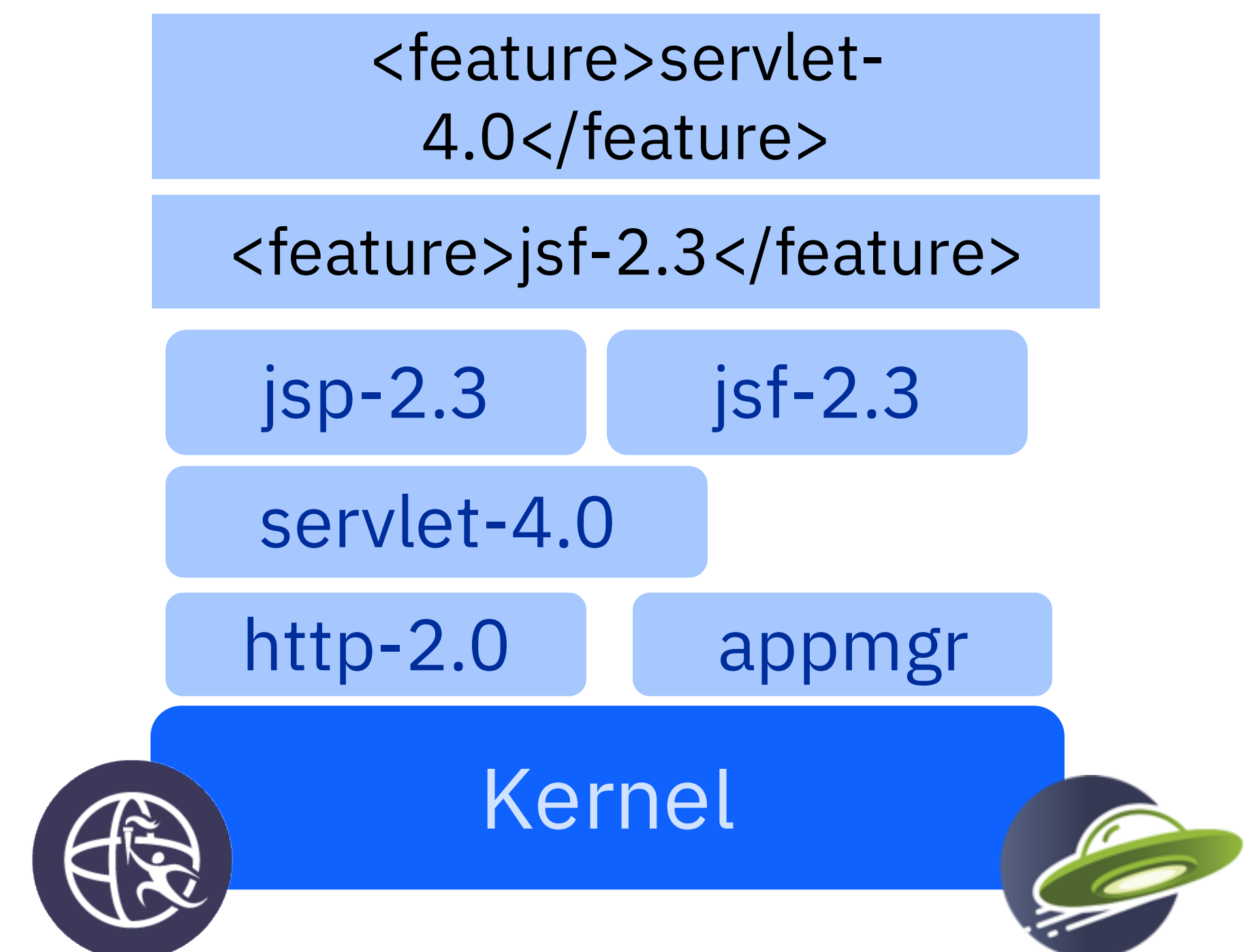
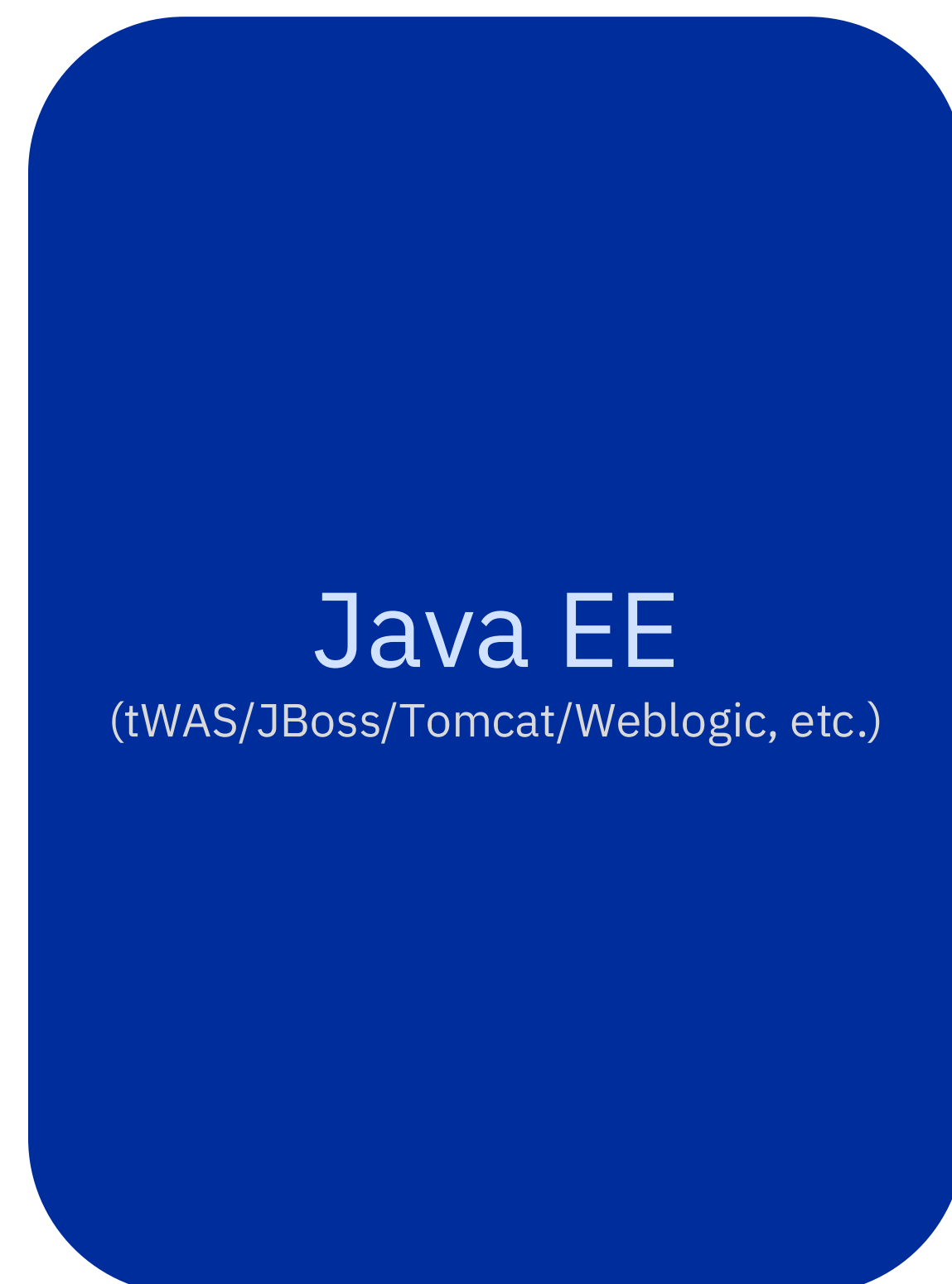
Uruchamiaj tylko to, czego wymaga Twoja aplikacja.

Tradycyjny monolityczny serwer aplikacji a modułowa architektura Liberty

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

Liberty: zaczynając od lekkiego kernela, 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

2. Niższe koszty operacyjne

Mniej pamięci, wysoka przepustowość, mniej instancji i mniej licencji

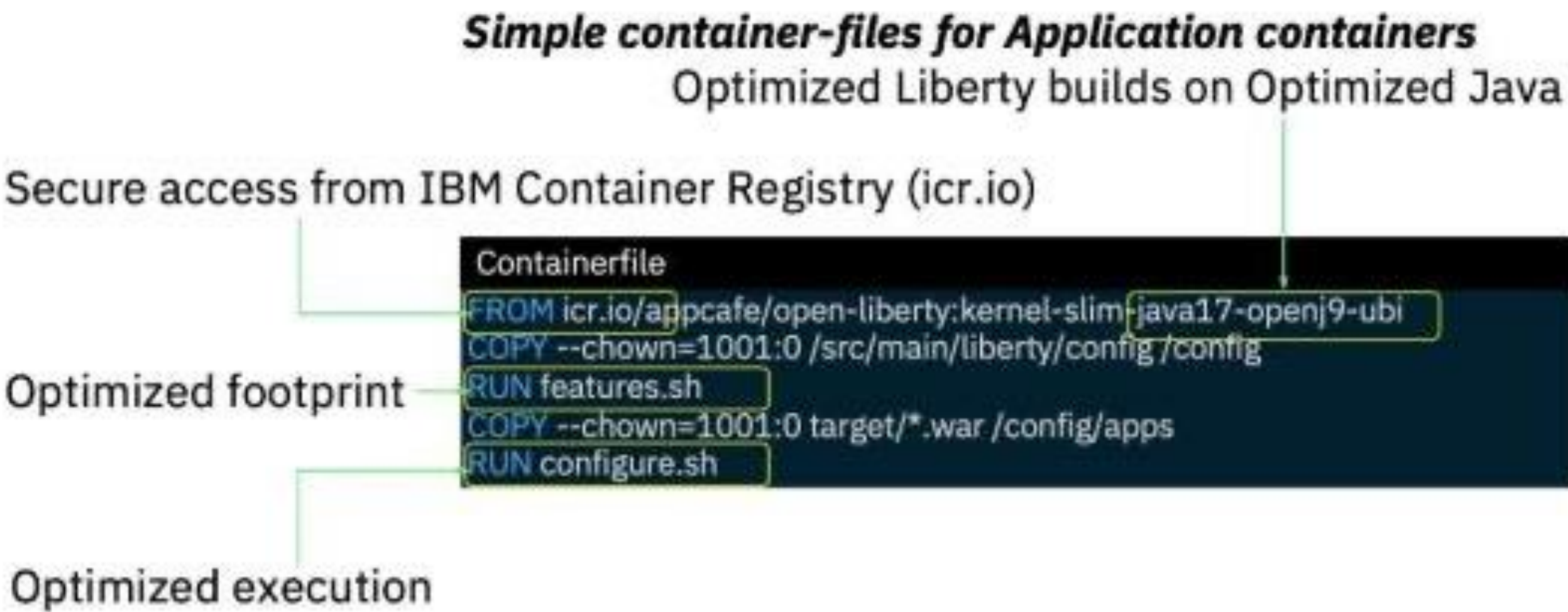
Prosta budowa w odpowiednim rozmiarze

Tworzenie aplikacji

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

Tworzenie kontenera

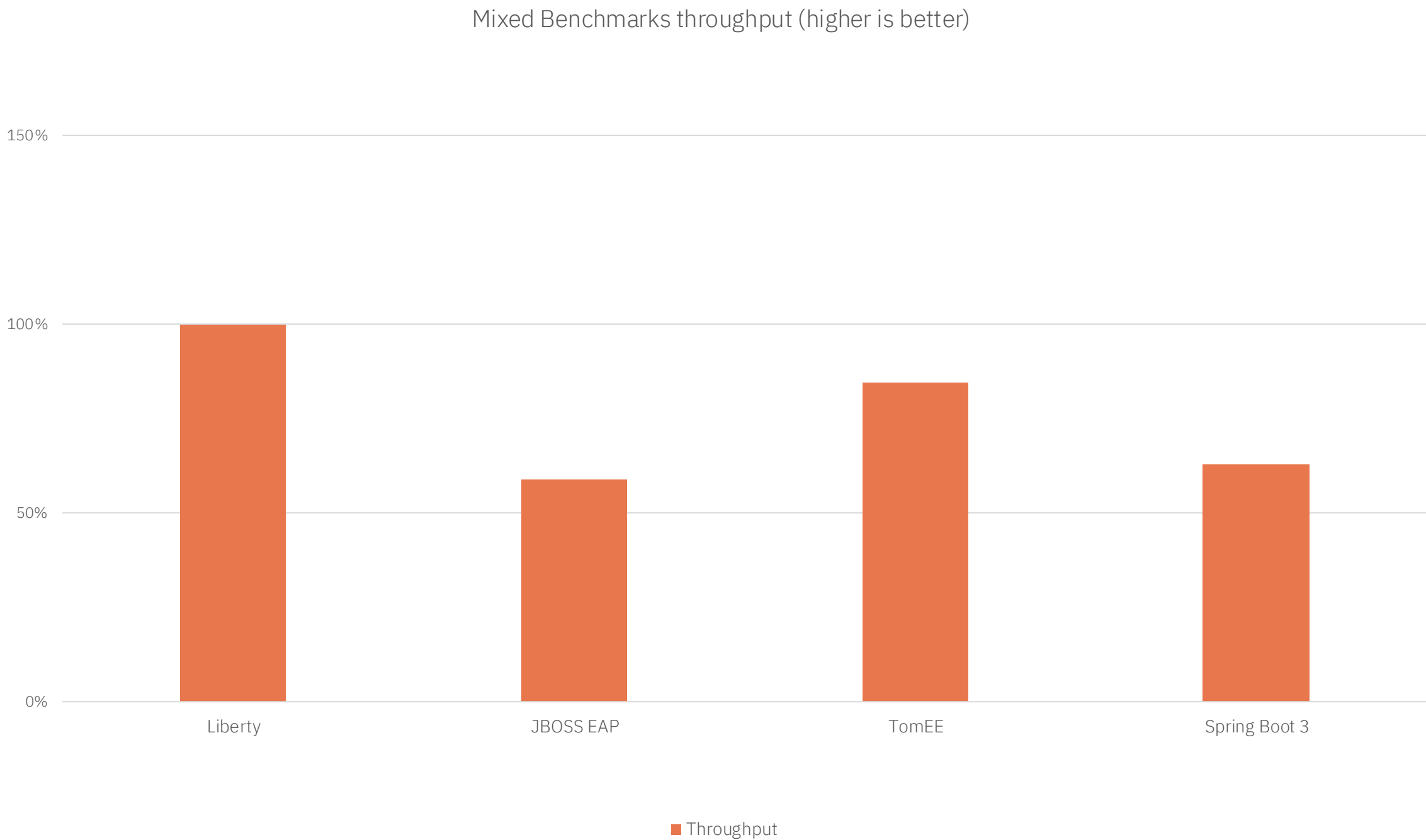
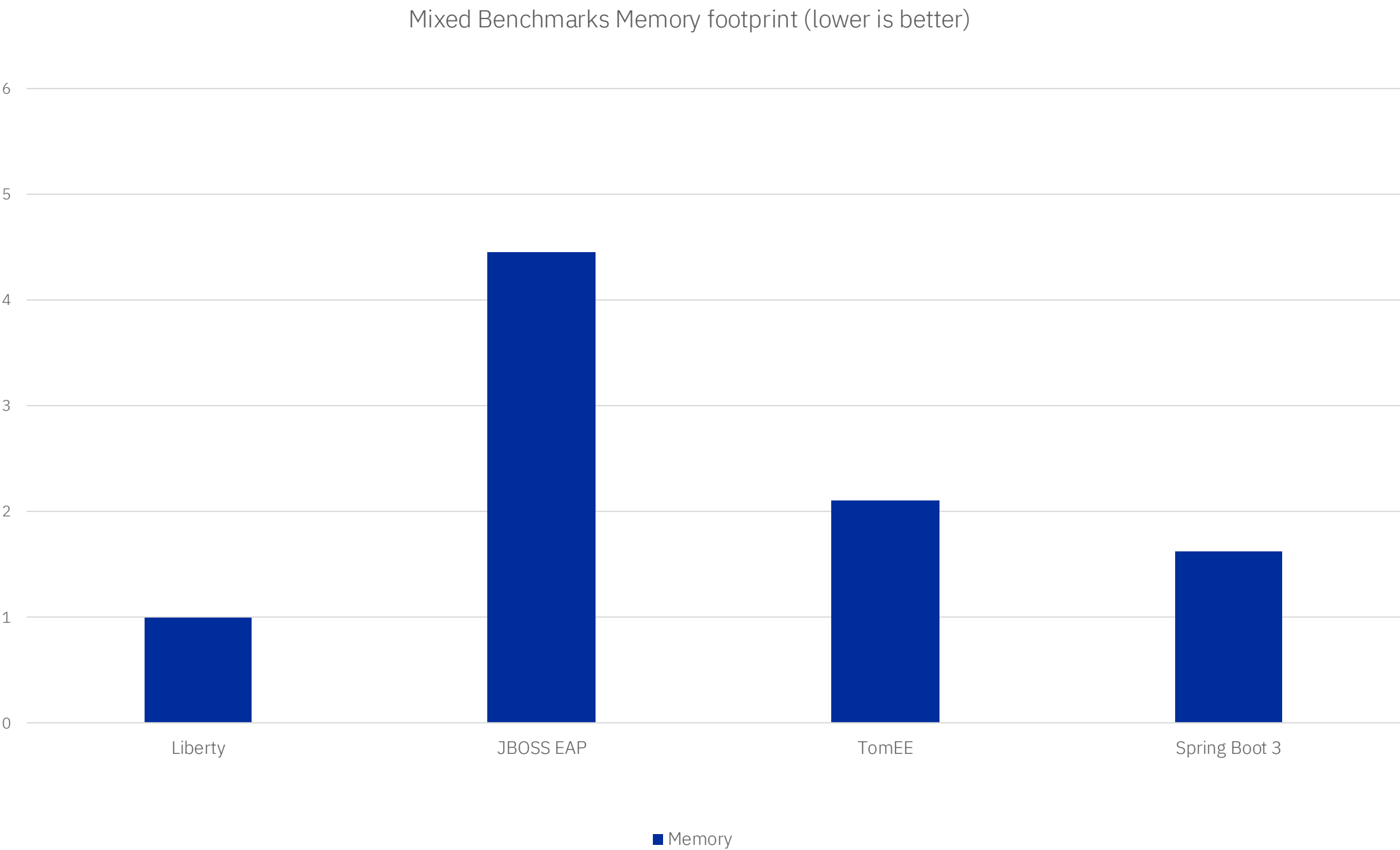
- Wiodące podejścia do tworzenia kontenerów - Dockerfile, Cloud Native Buildpack, Source-2-Image
- Certyfikowane obrazy Liberty opublikowane w IBM Container Registry



Package	Size on Disk	Memory
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

3. Ciągłe dostarczanie

Niskie koszty utrzymania, zerowy dług technologiczny,
uwzględnione poprawki bezpieczeństwa

and

4. Zero migracji

Eliminacja migracji z wersji do wersji

Ciągłe dostarczanie Liberty

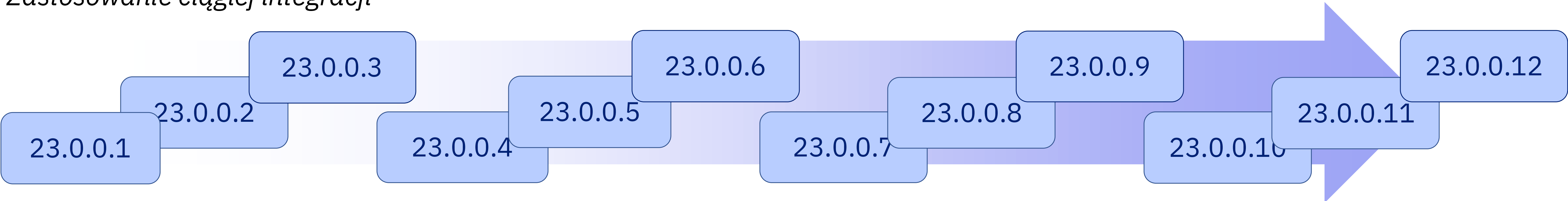
Architektura „zerowej migracji” Liberty sprawia, że wdrożenie nowej wersji jest proste

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
- 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

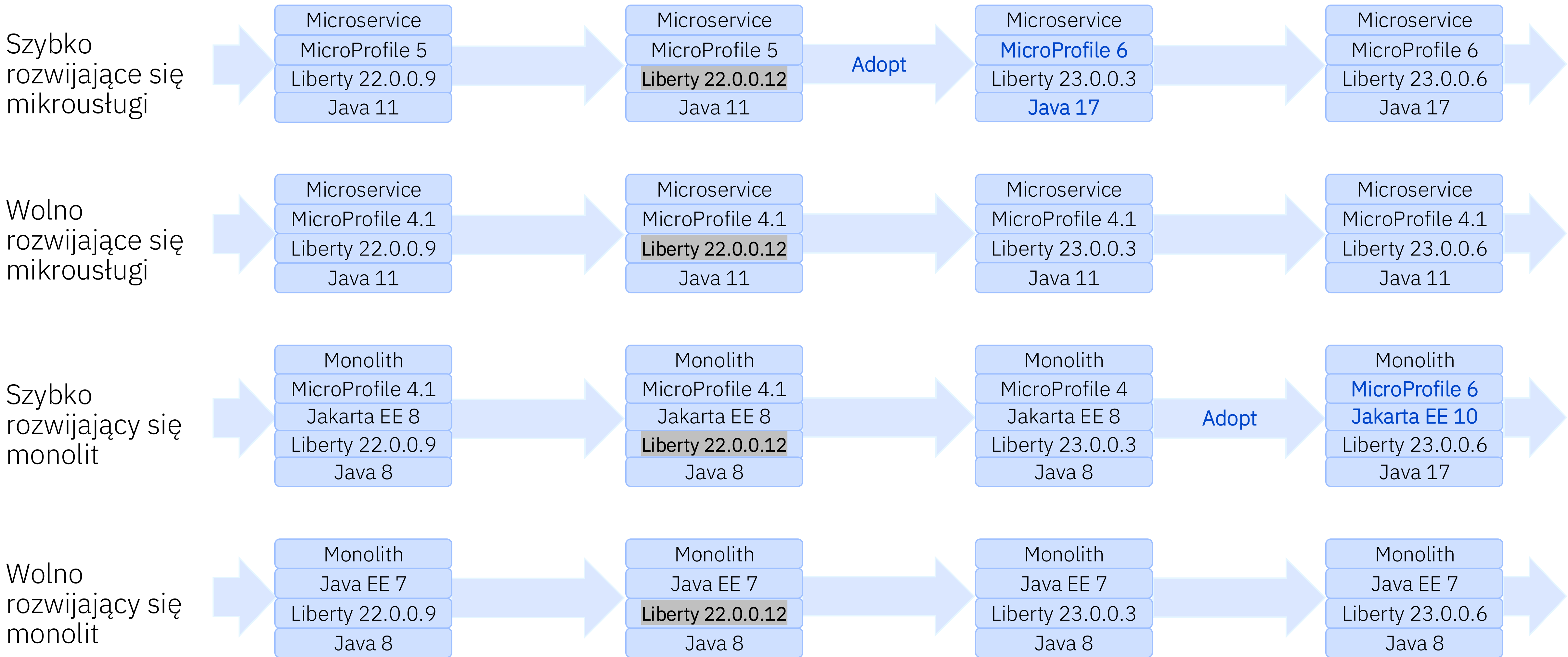
Liberty Zero Migracji

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



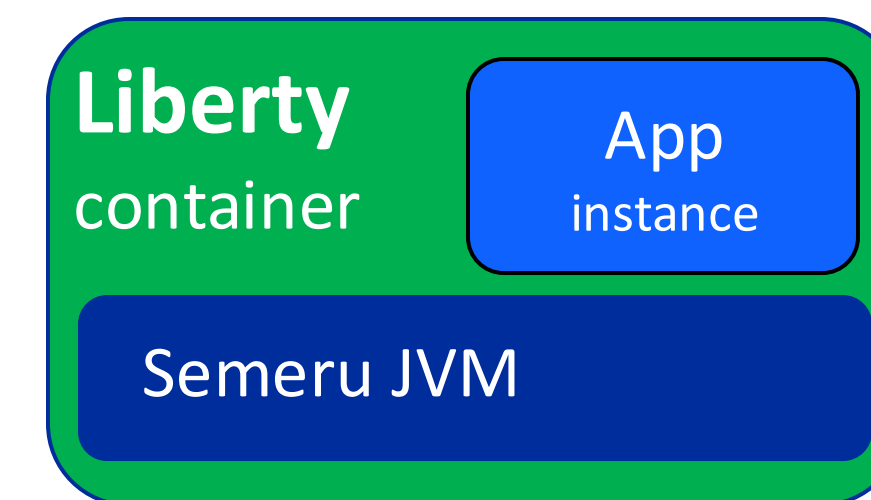
- Jakarta EE 10
- MicroProfile 6

5. Zoptymalizowane pod Kubernetes

Auto-tuning, łatwy build-deploy dla kontenerów

Pełne zastosowanie innowacji Java dla chmury

Optymalizacja kontenerów w chmurze na poziomie całego stosu Java



Proste container-files dla kontenerów aplikacji

Optimized Liberty budowane na Optimized Java

Bezpieczny dostęp z IBM Container Registry (icr.io)

Zoptymalizowany
ślad

Zoptymalizowane
wykonanie

Containerfile

FROM icr.io/appcafe/open-liberty:kernel-slim-java17-openj9-ubi

COPY --chown=1001:0 /src/main/liberty/config/config

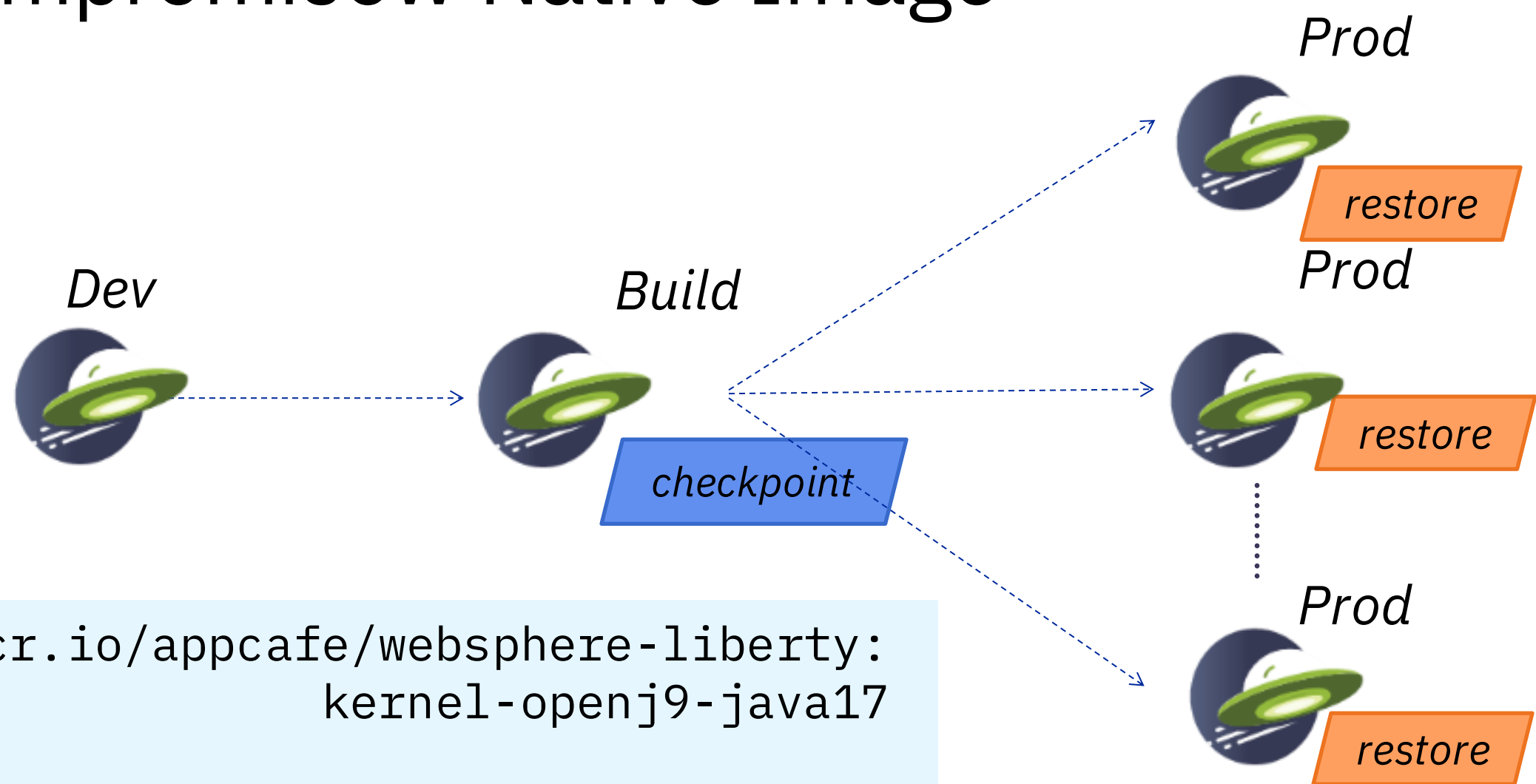
RUN features.sh

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 żadnych kompromisów Native Image



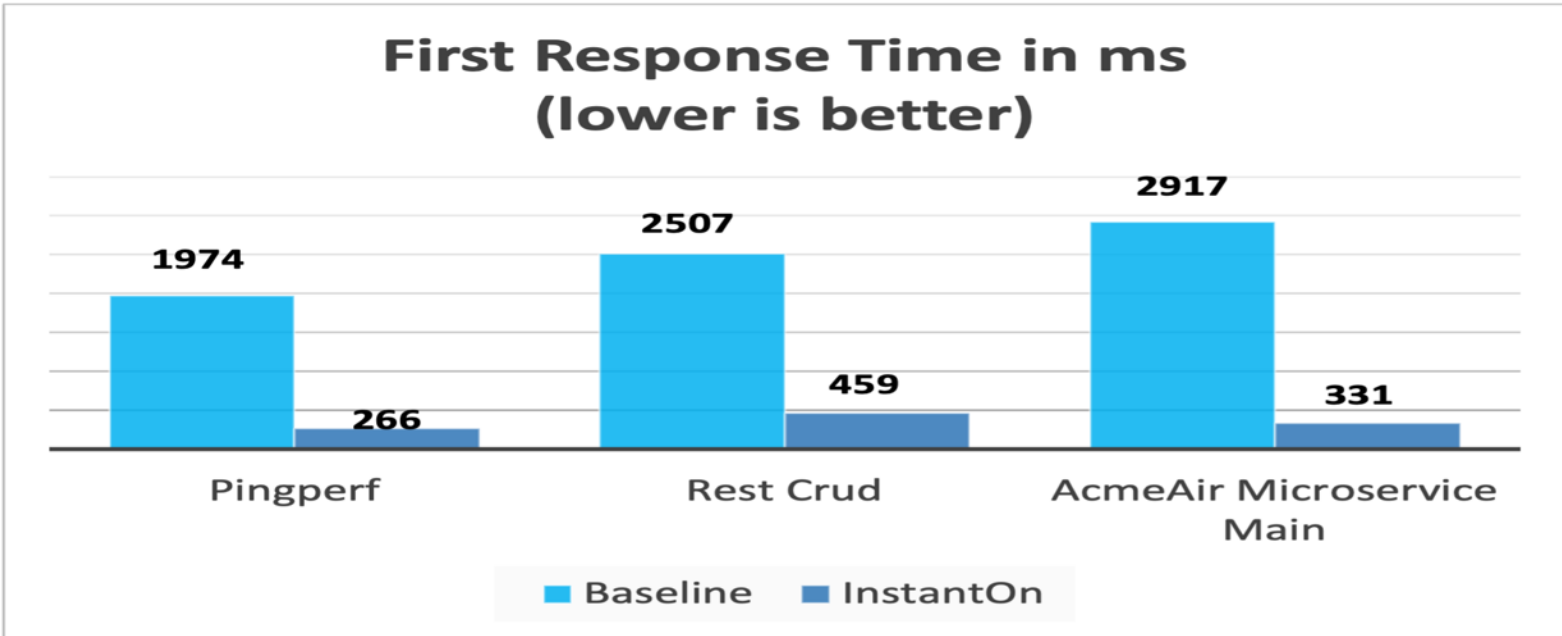
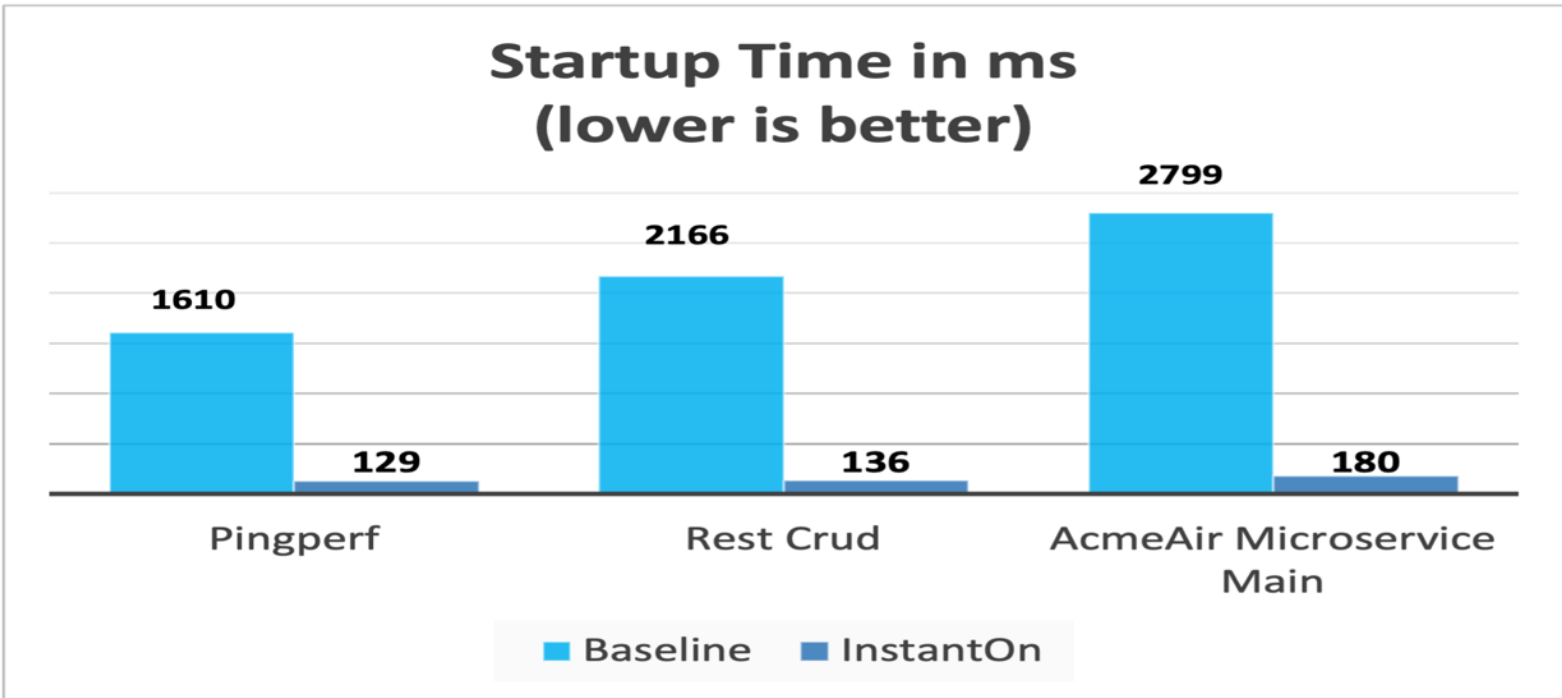
```
FROM icr.io/appcafe/websphere-liberty:
    kernel-openj9-java17

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

RUN features.sh

COPY target/*.war /config/apps/

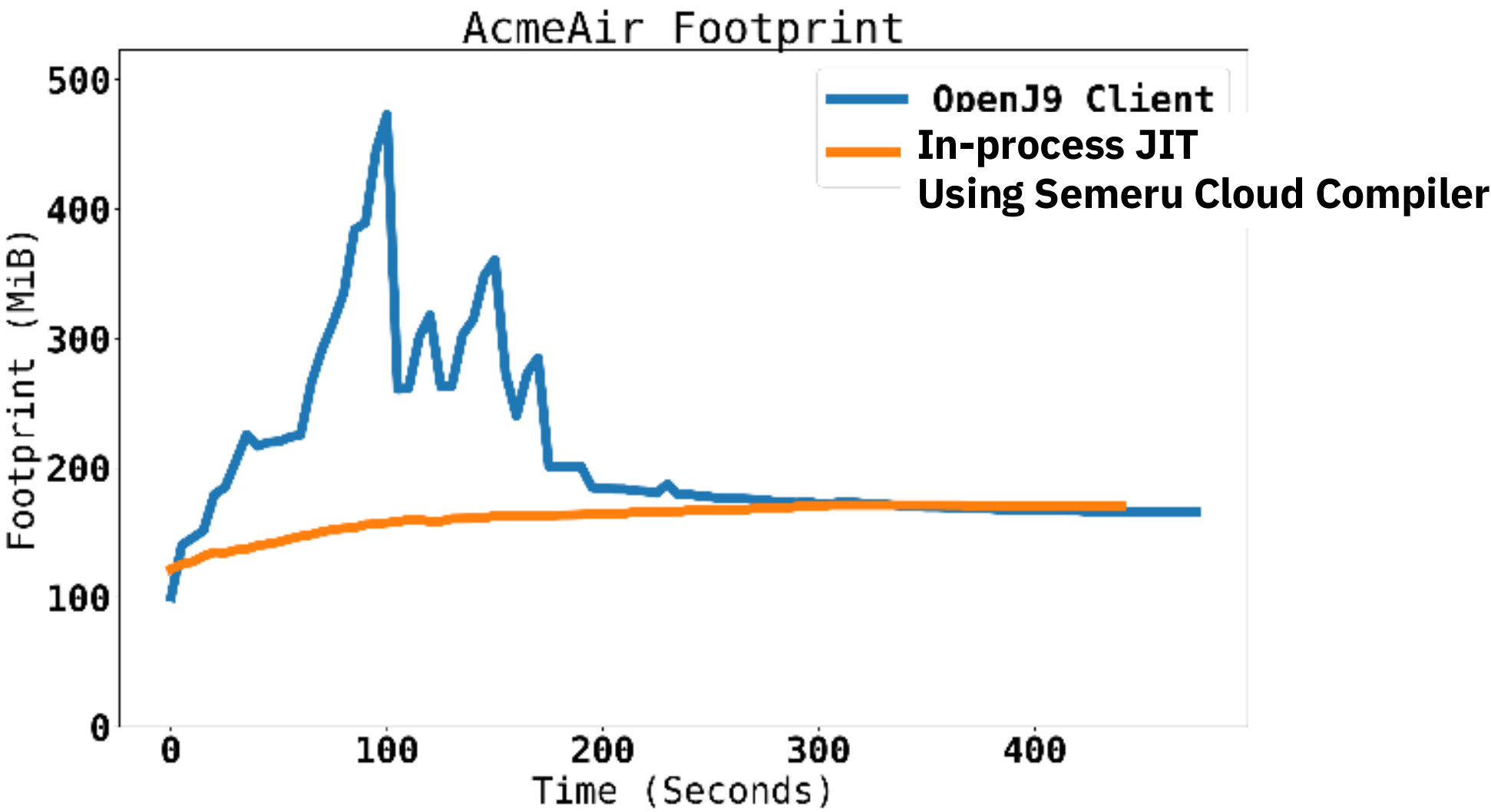
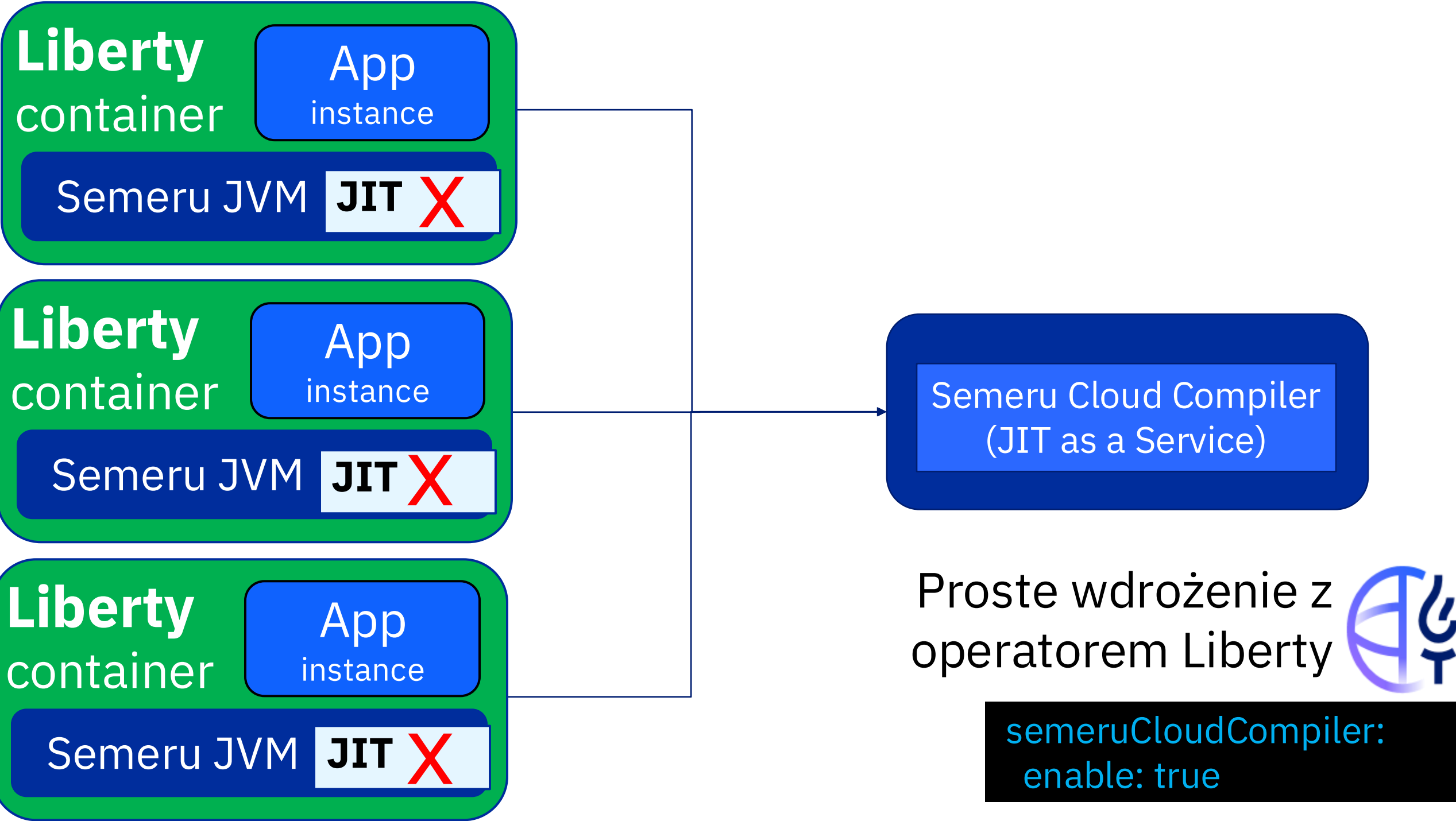
RUN configure.sh
RUN checkpoint.sh afterAppStart
```



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



Service	Memory limit w/o SCC Server	Memory limit with SCC Server	Saving
Auth	1,050 MB	750 MB	300MB
Booking	3,300 MB	2,400 MB	900MB
Customer	1,650 MB	1,050 MB	600MB
Flight	2,250 MB	1,250 MB	1,000 MB
Main	600 MB	450 MB	150MB
Total	8,850 MB	5,900 MB	2,950 MB

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
```

* Przykłady: Integracja zarządzania certyfikatami z OCP, delegowanie SSO

Operator

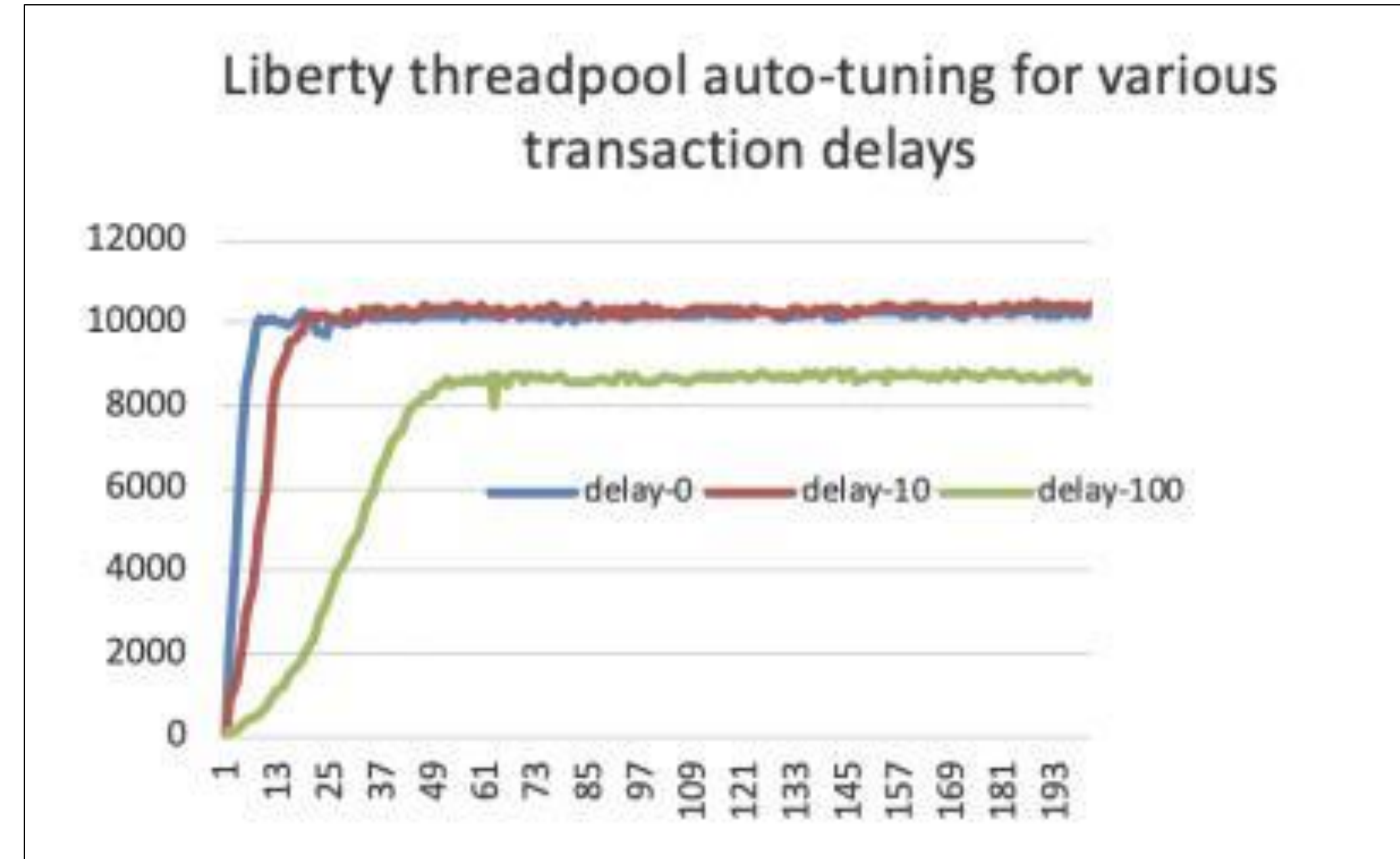
Custom Resource Definition (CRD)

- Definiuje nowy interfejs API
 - Rzeczywisty obiekt w nowym API
- Płynna integracja z istniejącym API
 - Wsparcie dla Kubectl

```
# Generated by IBM TransformationAdvisor
kind: WebSphereLibertyApplication
apiVersion: liberty.websphere.ibm.com/v1
metadata:
  name: plantsbywebsphereee6
spec:
  replicas: 1
  applicationImage: ''
  expose: true
  volumeMounts:
    - mountPath: /config/variables
      name: plantsbywebsphereee6-secret
      readOnly: true
  volumes:
    - name: plantsbywebsphereee6-secret
      secret:
        secretName: plantsbywebsphereee6-secret
  envFrom:
    - configMapRef:
        name: plantsbywebsphereee6-config
  manageTLS: false
  license:
    accept: true
```

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

Interfejsy API neutralne dla dostawców

Mikroustugi i natywne dla chmury interfejsy API dla przedsiębiorstw wolne od ograniczeń związanych z dostawcami

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

- Tworzenie nowych otwartych mikroustug natywnych dla chmury za pomocą MicroProfile, wykorzystując istniejące umiejętności i zasoby Java EE/Jakarta EE
- Modernizacja istniejących aplikacji 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

The screenshot displays the 'demo-devmode' application interface. The top section shows the 'server.xml' file in the 'EXPLORER' view, with the 'main' directory expanded. The 'server.xml' file is highlighted, showing its contents in the 'OUTLINE' view. The file is an XML configuration for a Liberty server, including features like jaxrs-2.1, jsonp-1.1, cdi-2.0, mpMetrics-2.0, and mpConfig-1.3. The bottom section shows the terminal output, which includes information about the Liberty container, port mappings, and the successful compilation of the application.

```
server.xml — demo-devmode
```

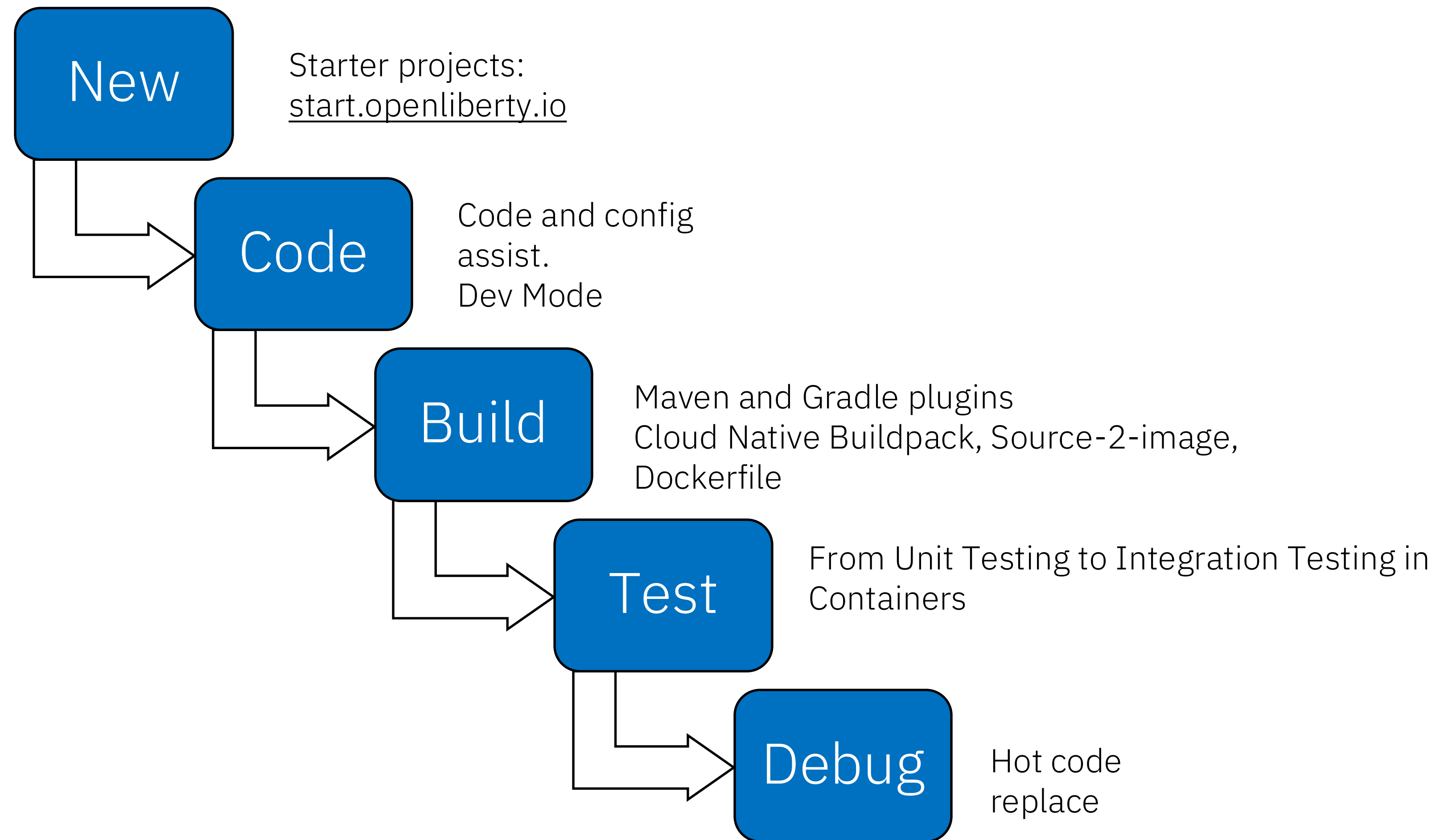
```
src > main > liberty > config > server.xml
```

```
1 <server description="Sample Liberty server">
2   <featureManager>
3     <feature>jaxrs-2.1</feature>
4     <feature>jsonp-1.1</feature>
5     <feature>cdi-2.0</feature>
6     <feature>mpMetrics-2.0</feature>
7     <feature>mpConfig-1.3</feature>
8   </featureManager>
9
10  <webApplication location="${artifactId}.
11
12  <mpMetrics authentication="false"/>
13
14  <!-- tag::logging[] -->
15  <logging traceSpecification="com.ibm.ws.
16  <!-- end::logging[] -->
17
18  <httpEndpoint host="*" httpPort="${defal
19  httpsPort="${default.https.port}" ic
```

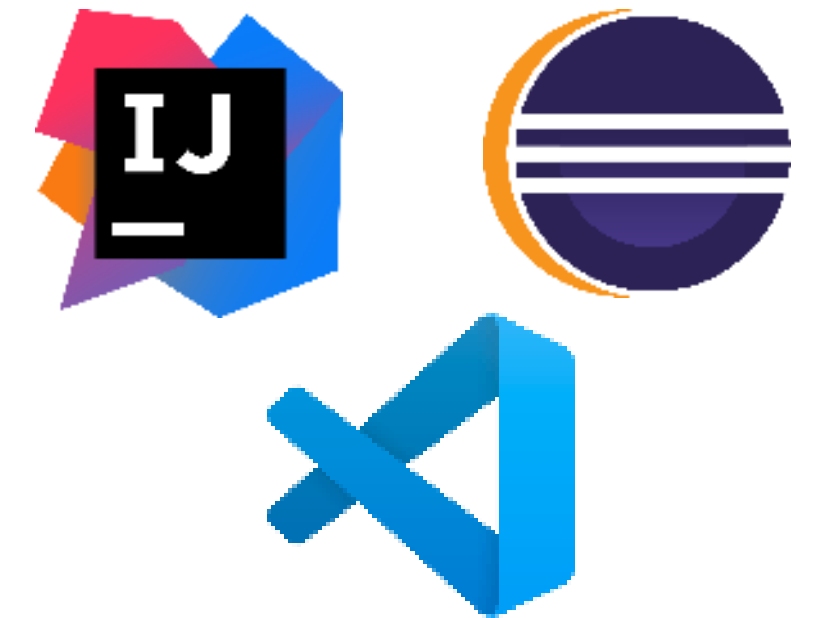
```
[INFO] *****
[INFO] *      Liberty is running in dev mode.
[INFO] *      To run tests on demand, press Enter.
[INFO] *      To rebuild the Docker image and restart the container, type 'r' and press Enter.
[INFO] *      To stop the server and quit dev mode, press Ctrl-C or type 'q' and press Enter.
[INFO] *
[INFO] *      Liberty container port information:
[INFO] *      Internal container HTTP port [ 9080 ] is mapped to Docker host port [ 9080 ]
[INFO] *      Internal container HTTPS port [ 9443 ] is mapped to Docker host port [ 9443 ]
[INFO] *      Liberty debug port mapped to Docker host port: [ 7777 ]
[INFO] *
[INFO] *      Docker network information:
[INFO] *      Container name: [ liberty-dev ]
[INFO] *      IP address [ 172.17.0.2 ] on Docker network [ bridge ]
[INFO] *****
[INFO] Source compilation was successful.
[INFO] Tests compilation was successful.
[INFO] [AUDIT   ] CWWKT0017I: Web application removed (default_host): http://c1bf2d4d704a:9080/
[INFO] [AUDIT   ] CWWKZ0009I: The application demo-devmode-maven has stopped successfully.
[INFO] [AUDIT   ] CWWKT0016I: Web application available (default_host): http://c1bf2d4d704a:9080/
[INFO] [AUDIT   ] CWWKZ0003I: The application demo-devmode-maven updated in 1.157 seconds.
```

Wydajność dewelopera

Szybki iteracyjny rozwój w wybranym IDE



Liberty Tools



TRY IT



