WebSphere ReVisit資料

2025/07/14
2025/10/06 更新

Agenda

1. WAS再訪の背景(ReVisitの目的)
2. Javaの進化とサポート状況
3. 業界での採用状況と動向
4. “Javaは古い”という誤解と構成の変化
5. Libertyという選択肢
6. まとめと次のステップ
7. 参考

- 「まだ5年ある」問題
- WASが注目される理由
- Java8問題を支えるオファーリング
- 商談前ヒアリングリスト／トークスクリプト
- コミュニティ加入促進

用語説明:

■ そもそもJavaやWASは何？

– Java

- ここではJava SE(standard edition)。Javaアプリケーション を開発・**実行するための土台**となる仕様やツール群
- Sun Microsystems が中心に開発、2010年に Oracle に買収。現在は Java Community Process（JCP） を通じて、OpenJDK（GPLライセンスのオープン実装）が公式リファレンス実装として運営。2017年には Eclipse Foundation に移管され、名前も Jakarta EE に変更。

– Java EE/Jakarta EE

- Java Enterprise Editionの略。JavaSEが導入された上で動作できるフレームワーク。エンタープライズ向けAPI群。

– ライブラリ

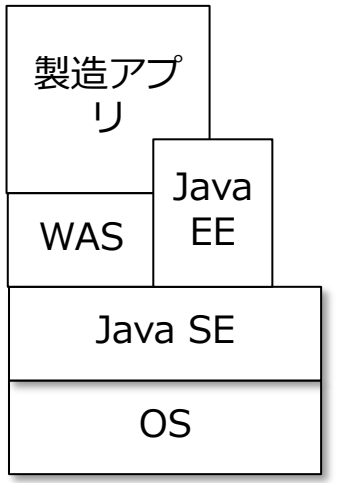
- いつ使うか自分で決める、小さな機能単位のプログラムのパーツ。特定の機能（例：文字列処理、日付計算、HTTP通信など）を提供。開発者の構成に応じ、呼び出し機能を発揮させる。

– フレームワーク

- 設計・処理の流れを定めてくれる枠組み。流れは決まっているので、そこに開発者がプログラムを埋め込む。

– WAS (WAS traditional = tWAS, WAS Liberty)

- IBM Websphere Application Serverの略。製品名。Javaを使い**Web経由で動的アプリの実行に必要な仕組み**を提供するソフトウェア。



用語説明:

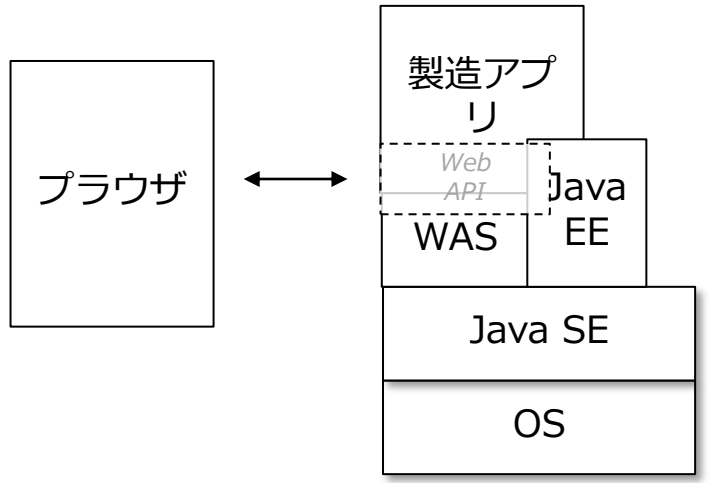
■ APIって？

– API

- ソフトウェア同士が機能やデータをやり取りするための「約束事（ルール・仕様）」。開発者向けに公開され、「何を渡せば何が返ってくるか」を示すインタフェースの仕様と実装一式を指します。
- プログラム用API（ライブラリ／フレームワークのAPI）は、複雑な処理を簡単に呼び出せるように抽象化された関数。

– Web API

- HTTP（GET, POST, PUT, DELETE）経由でリモートのサーバー機能を呼び出すためのインタフェース仕様。Web APIで 「クライアントがサーバーに何をしてほしいか」を伝える命令 です。命令を受けてプログラム(ここではJavaプログラム)が動き出し処理した結果を返します。



Revisitにおける目的

1. Java8サポート終了に対して、既存のWAS利用の維持とアップセル
   - WAS継続利用のためのLiberty化移行推進
   - クラウドファースト企業への訴求
   - 開発・検証環境利用としての訴求

   →最新Liberty ライセンス(Jsphere)への移行(買い替え)

2. お客様のご利用環境・課題を把握することで新たなビジネス機会を創出
   - アプリケーションモダナイズ → TAや生成AIを利用した移行や開発サイクルの効率化
   - 運用の高度化・自動化 → Instana/Trubonomicの提案に繋げる

3. 既存のWASでないJava環境をお持ちのお客様の獲得
   - Java8の継続利用が必要なお客様へのアプローチ
   - Java8からバージョンアップを低コストにしたいお客様へのアプローチ

   →WASへのWin back、最新Libertyライセンス(Jsphere)やxCAにて既存Java8環境の最適化

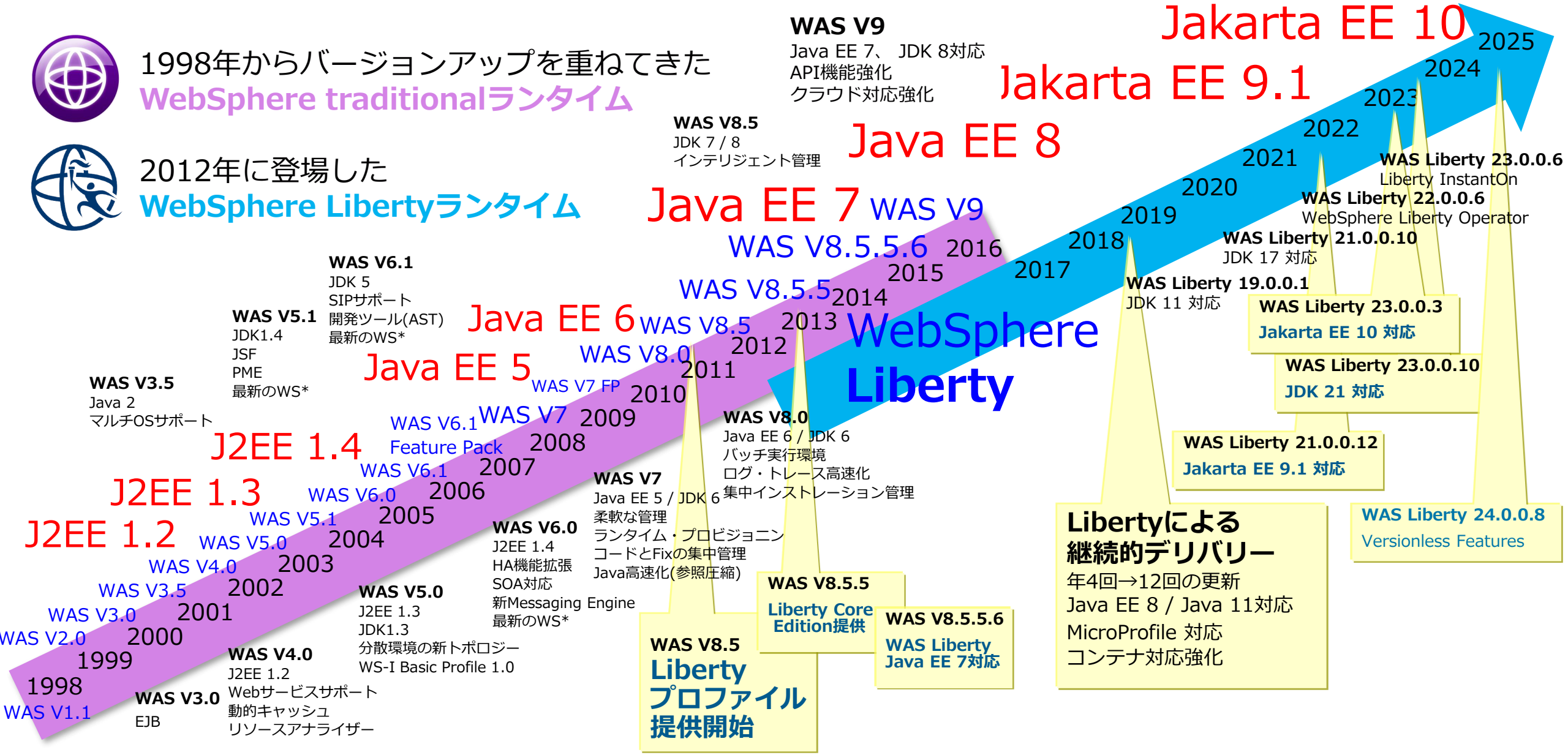
WebSphereは今年で27周年

これを見せてWASは終わると風評を流し乗り換えを進めている事案が発生しています。

WAS Liberty 24.0.0.8
フィーチャーのバージョンレス機能

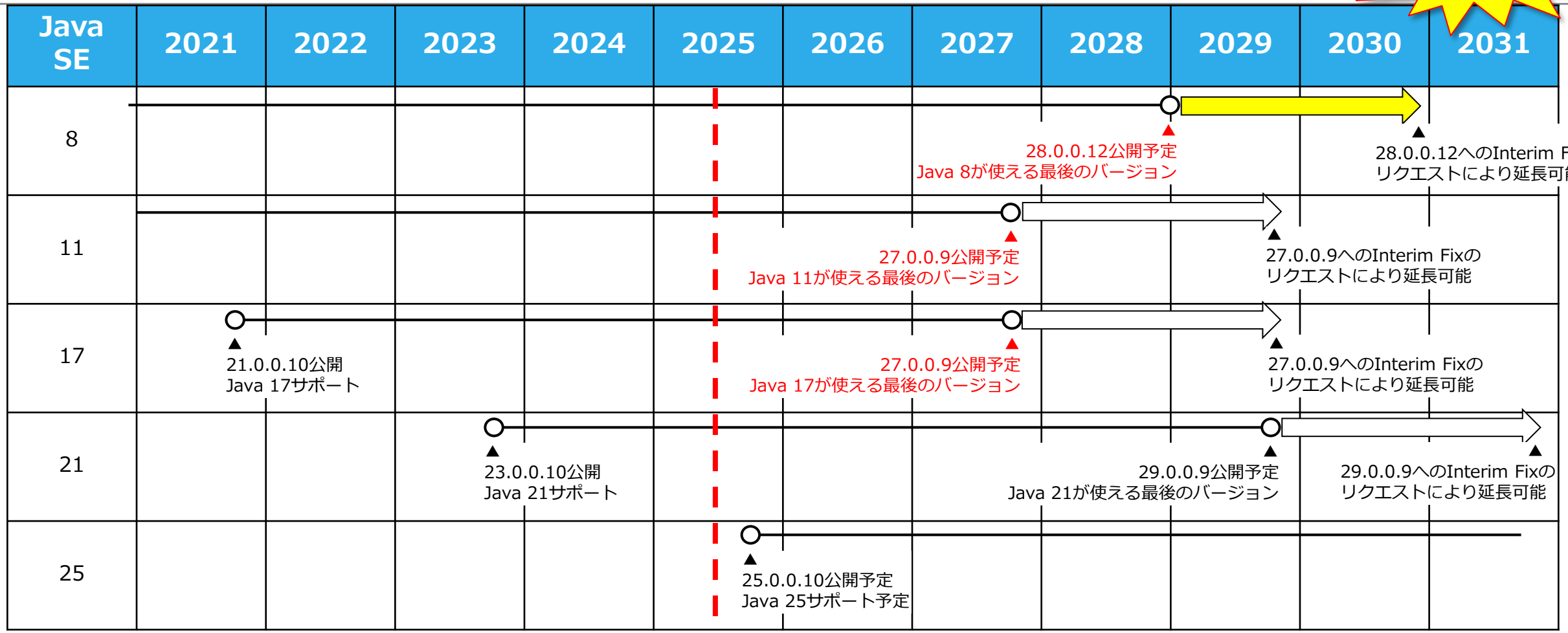
1998年からバージョンアップを重ねてきた
WebSphere traditionalランタイム

2012年に登場した
WebSphere Libertyランタイム



Java SEとLibertyの対応バージョンの現状(最新)

修正あり



- LibertyのJava 8サポートは2028年公開予定の28.0.0.12版までの予定
- LibertyのJava 17サポートは2027年公開予定の27.0.0.9版までの予定
- LibertyのJava 21サポートは2029年公開予定の29.0.0.9版までの予定

2014年に公開されたJava8も、いよいよサポートが順次終了していきます

	Release date	End of public updates	End of extended support(有償)
Java SE 8	2014/3/18	2019/4. Oracle 2026/11 Eclipse Temurin 2026/11 Red Hat 2026/11 Azul 2030/12 Amazon Corretto **2025/4. IBM Java SDK** **2026/11 IBM Semeru Runtimes**	2030/12 Oracle 2030/12 Azul 2031/3 BellSoft Liberica

終了宣言を取りやめるところもありと、煩雑な状況です。

オープンソースのライブラリやフレームワークの多くは、
Java 8に対してはセキュリティのアップデートや新機能を提供しなくなっています。

Spring	**Jakarta EE 9+**
Kafka	MicroProfile 5+
Spark	
Hibernate	
+100s more	

フレームワークはどう言ったものをお使いか把握が必要です。

今後も機能追加・メンテナンスを続けるシステムで、**サポートの無いJava 8**を使い続けることは、現実的ではない状況になってきています

Java8に関するサポート終了状況

Java8仕様を活用している製品/OSSのサポート終了が増加している。

Java8 EoSに向けての各社対応状況

JBoss EAP8（最新版）はJava8は対応していない。

*お客様はEAP7使っている？

Oracle WebLogic も最新版はJava8は対応していない。

Jakartaは未対応

Fujitsu Enterprise Application Platform :Liberty

Fujitsu InterStage :Traditional

IBM Liberty 26.0.X代は2026年9月リリース版でサポート終了。

IBM Tranditional WASも2030年でサポート終了。

最新技術の採用ができない：利用者のための実装要望にお応えすることが難しい。

技術負債の維持：古い技術者を抱え続けなくてはいけない。

市場・技術周辺状況

エンジン以外の周辺ライブラリやフレームワーク自体もJava8 Supportが終了となっている。

技術者が少なくなってきている。

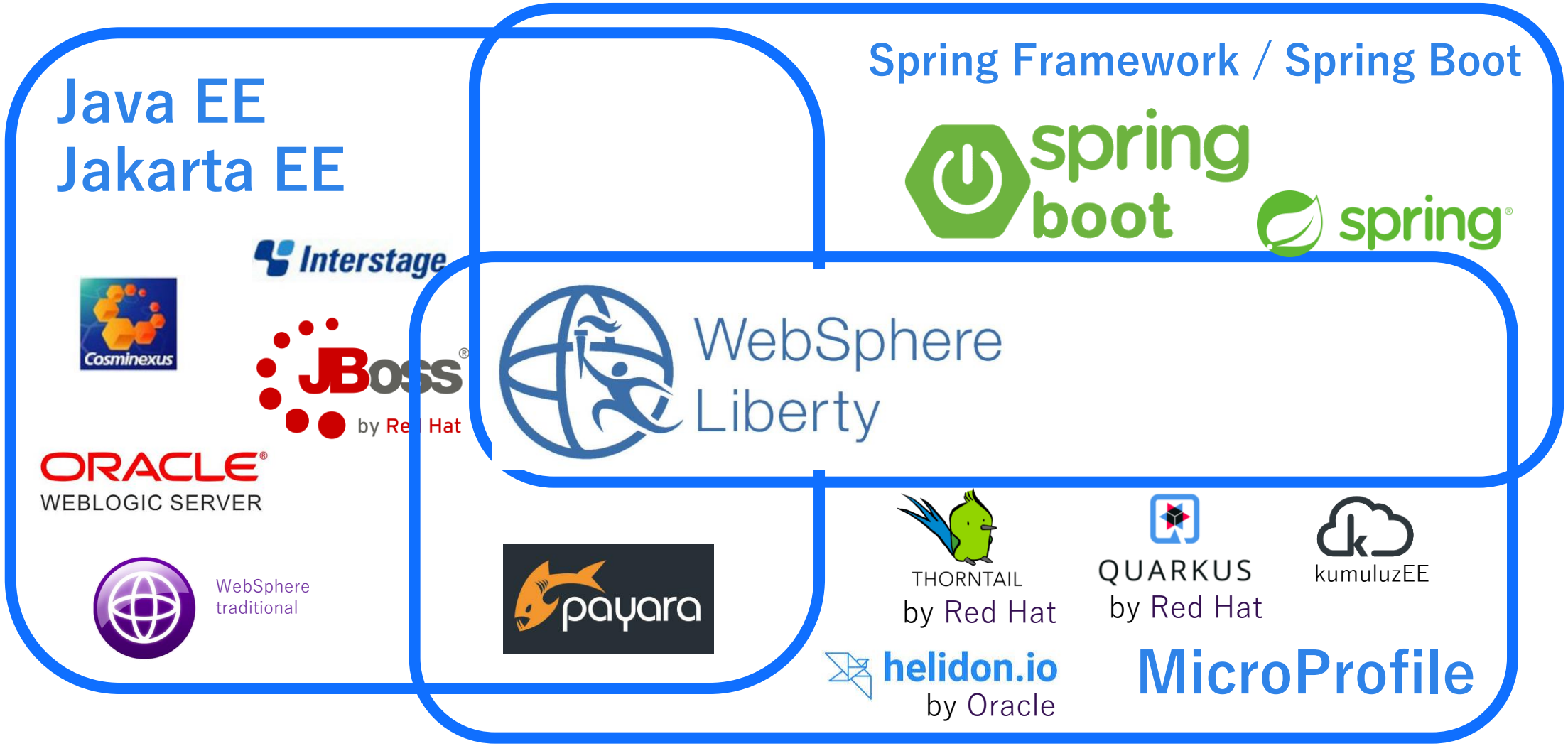
Libertyとtraditional：用途に応じたランタイムの選択

- **WAS traditional**（WebSphere traditional / tWAS / 従来型WAS）
  - **既存資産の活用**を目的としたランタイム
    - 旧WASでおこなっていた**従来の運用を継続**したいお客様
  - Libertyで対応していない**旧API**を使用しているアプリケーションの実行環境として
    - JAX-RPCやEntity Beanなど
  - **今後，新機能の実装や新しい仕様への対応は行われない**
    - Java EE 7 / Java 8に対応した2016年出荷のV9.0が**最後のバージョン**

- **WAS Liberty** (WebSphere Liberty)
  - モダンなアプリケーション開発・サーバー運用に対応した**新時代のランタイム**
    - 軽量さを活かしたAgile開発やCD（継続的デリバリー）
    - ツールによる運用の自動化・DevOps（Platform as a Code / Immutable Infrastructure）
  - WAS traditionalとは，運用に対する設計思想が根本的に異なる
  - **クラウド**での使用やコンテナ環境，リソースの限定されたIoT環境にも最適
  - 今後も**継続して進化**を続ける

Java 8や最新Java EEだけでなく、MicroProfileにも対応しているWebSphere Libertyが、移行先としてお勧めです

エンタープライズJavaの業界動向



各社のモダナイズへの取り組み

開発者のスキルや既存アプリを活用したままクラウドネイティブへの対応ができるのはIBMのWebSphereだけ

従来型ランタイム

JBoss by Red Hat
JAKARTA EE
Java EE
MICROPROFILE
*1

ORACLE WEBLOGIC SERVER
Java EE

WebSphere traditional
Java EE

移行しやすい

クラウドネイティブ対応・次世代ランタイム

QUARKUS by Red Hat
MICROPROFILE

helidon.io by Oracle
MICROPROFILE

WebSphere Liberty by IBM
MICROPROFILE
Java EE
JAKARTA EE

*1 限定的にサポート

フレームワーク(Java(Jakarta) EE)による提供機能とJava SEの比較

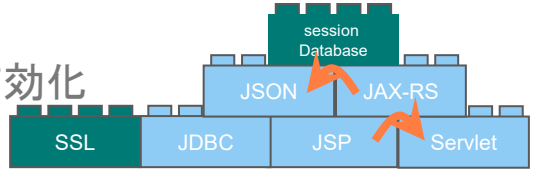
機能カテゴリ	EEで提供される機能	Java SEのみではどうなるか	企業利用での影響
Webアプリ機能	@WebServlet、 JSP、 JSF	Servletコンテナがない	Web画面が作れない／HTTP応答できない
データベース連携	JPA (@Entity、@PersistenceContext)	ORMがない／自前実装が必要	DBアクセスが困難／SQL直書き地獄
トランザクション管理	JTA、 @Transactional	トランザクション処理なし	データ整合性が崩れる（例：入金処理で不整合）
依存性注入（DI）	CDI、 @Inject、 @Named	依存性の自動解決不可	クラス間の結合が強く、テストや拡張が困難
セキュリティ(認証関連)	Java EE Security (@RolesAllowed etc)	手作業で認証／認可	権限処理が複雑化しミスを誘発
メッセージング	JMS (@MessageDriven)	MQ連携なし	非同期通信が実現できない／リアルタイム処理困難
Webサービス	JAX-RS (@Path)、 JAX-WS	REST/SOAPが使えない	外部API連携ができない／他社システムと非連携
ライフサイクル管理	@PostConstruct、@PreDestroy	無効（呼ばれない）	初期化・終了処理が漏れる／リソースリークの原因(突然停止リスク)

※Microprofileもまた、呼び出し名は違うが、同じような機能を持っている

サーバー環境がコードとして定義されているため、超高速・軽量です
クラウド環境での利用にも適しています

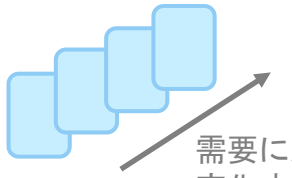
1 モジュール化されたランタイム

- 機能をFeature として
  モジュール化
- 必要な機能だけを有効化
  自分のアプリにとって
  必要十分な環境を構築！



2 軽量、高速起動

- 数十MBのメモリ消費
- 100MB以下のランタイム
- わずか 数秒で 即時起動



需要に応じて
変化する環境に最適

3 シンプル構成 自動化/コンテナ化に最適

- 構成ファイルは１つだけ
- アプリケーションは
  配置するだけでデプロイ完了
- 公式コンテナイメージも
  RedHat UBIベースで毎月リリース

```
<featureManager>
    <feature>jsp-2.3</feature>
    <feature>jdbc-4.1</feature>
    <feature>jaxrs-2.0</feature>
    <feature>sessionDatabase-1.0</feature>
    <feature>ssl-1.0</feature>
</featureManager>
```

機能追加も再起動不要、動的に反映

4 新機能・欲しい機能がいま使える

- オープンソース OpenLiberty で開発
  https://openliberty.io/
- 市場の要求、技術動向を取り込み、毎月リリース
- Jakarta EE 8 も、世界最速でサポート

5 あなたのアプリに “ちょうどいい”

- 多用なフレームワークをサポート

クラウドネイティブ・アプリケーション

Eclipse MircroProfile 3、SpringBoot

ミッション・クリティカル・アプリケーション

Jakarta EE 8 FullProfile / WebProfile

✓従来型WebSphereからの国内移行実績も多数！

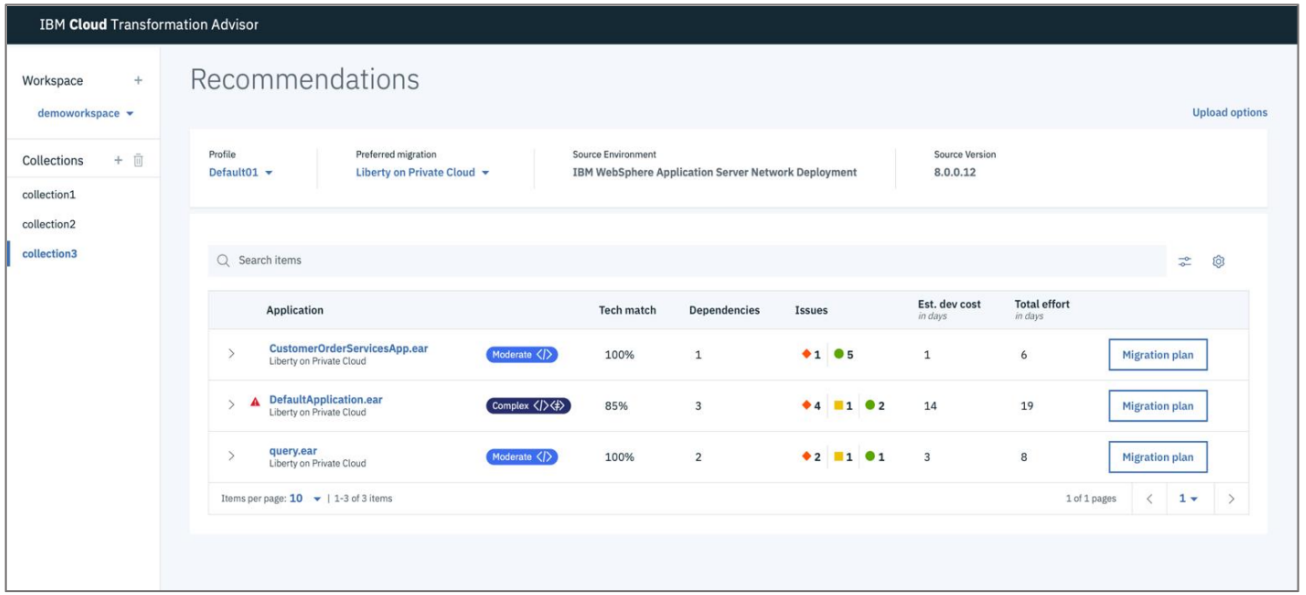
6 安心の ” ZERO Migration Policy ”

- 新しいバージョンの仕様が提供されても
  旧来バージョンのモジュールも提供を継続
- 構成ファイルのバージョンを変更しなければ
  古いバージョンのFeatureをそのまま利用可能

最適な移行ツール：WAS traditionalや他社製品からLibertyへの移行に

■ Transformation Advisor

- 既存アプリケーションサーバー環境/アプリをスキャン
  - WebSphere, WebLogic, JBoss をサポート
  - アプリケーションのみであれば 国産ベンダー環境でも
- マイグレーションの考慮点整理、レポート生成
- コンテナ化に必要な一連のファイルの自動生成
  - コンテナファイル, Liberty構成ファイル（server.xml)など
- OSSのRewriteを使用した自動修正に順次対応



Simple
そのままコンテナ化可能

Moderate
一部コード修正など対応が必要

Complex
一部アプリケーションの書直しが必要

事例：生成AIにより従来型WASからLibertyへ

Javaモダナイゼーション

PowerM

課題：

- **モダナイゼーション対応のリソース不足：**15年以上のWebSphere実績を持つrKubeは、EMEA地域でのモダナイゼーションとJavaアップグレードの需要が急増し、**Java開発のリソースが不足**。
- **生成AIソリューション信頼性の課題：**生成AIソリューションの導入を検討中で、**Javaモダナイゼーションの複雑さを理解**し、信頼性の高い生成AIソリューションがない。
- **開発ライフサイクル全体への支援が必要：**アプリケーションを迅速に理解し、最適なコード変換の提案、そしてモダナイゼーション過程でのコード変更の正確な検証の**開発ライフサイクル全体への支援**が必要。

ソリューション：

IBM watsonx Code Assistant(WCA)の導入で開発サイクルを加速:

- **コード生成**：WCAにより、レガシーJavaコードのモダナイゼーションや新規Javaアプリケーションの自動生成を実現し、**開発時間を大幅に短縮**。
- **テスト生成**：WCAがユニットテストを自動生成し、迅速な検証をサポート。**信頼性を確保して品質管理を強化**。
- **コード説明**：WCAがコードを分析し、各ファイルのクラスとメソッドを要約して提供。ドキュメント整備により**可読性が向上**し、**ナレッジシェアが促進**される。

“WCAにより、Javaモダナイゼーションがこれまで以上に簡単かつ迅速になり、品質も向上しました。この製品を他の方にもぜひお勧めしたいと思いますし、IBMの生成AIの今後に大いに期待しています。

— Walid Largou
CEO of rKube

従来のWebSphere
アプリケーションコードの **80%** を自動的にLibertyへ変換し、
最新アーキテクチャへモダ
ナイゼーション

Java 戦略のための費用対効果の高いオプション

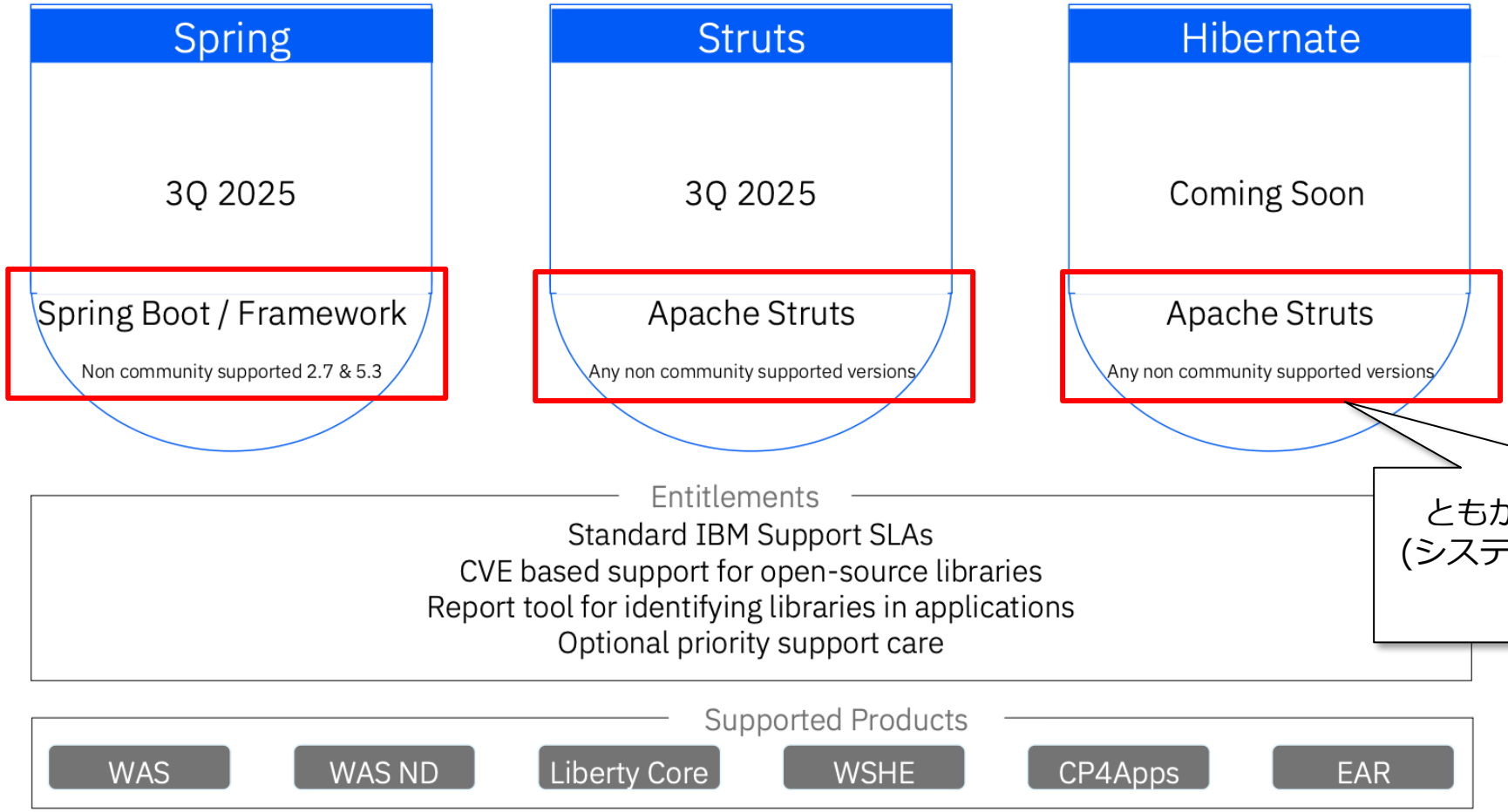
	IBM JSphere Suite for Java					
製品名	Enterprise Application Runtimes (EAR)			Enterprise Application Service for Java (EASeJ)	Cloud Pak for Applications (CP4Apps)	代替ソリューション (限定的な自動化)
コンポーネント	Open-Source Library Support for Java	Modernized Runtime Extension for Java (MoRE)	Liberty		Liberty	
モダナイゼーション戦略 →	Java 8 を維持	Java をアップグレード (WAS Admin)	クラウドに移行 + Java をアップグレード + コンテナを導入 + マイクロサービス	IBM マネージド・クラウドに移行 + Java をアップグレード	ハイブリッドクラウドに移行 + Java をアップグレード + コンテナを導入 + マイクロサービス	各種
時間の節約*	100%	95%	65%	80%	65%	なし
移行から得られる変革的なメリット	**該当なし**	**中**	**高**	**高**	**高**	**高**

自社資産のそれぞれのアプリケーションにとって合理的な戦略とオファリングを使用

* IBM Consulting のモダナイゼーション・エンゲージメント分析と、支援なしの 700 時間に基づく (アプリケーションあたり)

Open-Source Library Support for Java

Modernization Option 1: Support for open-source libraries

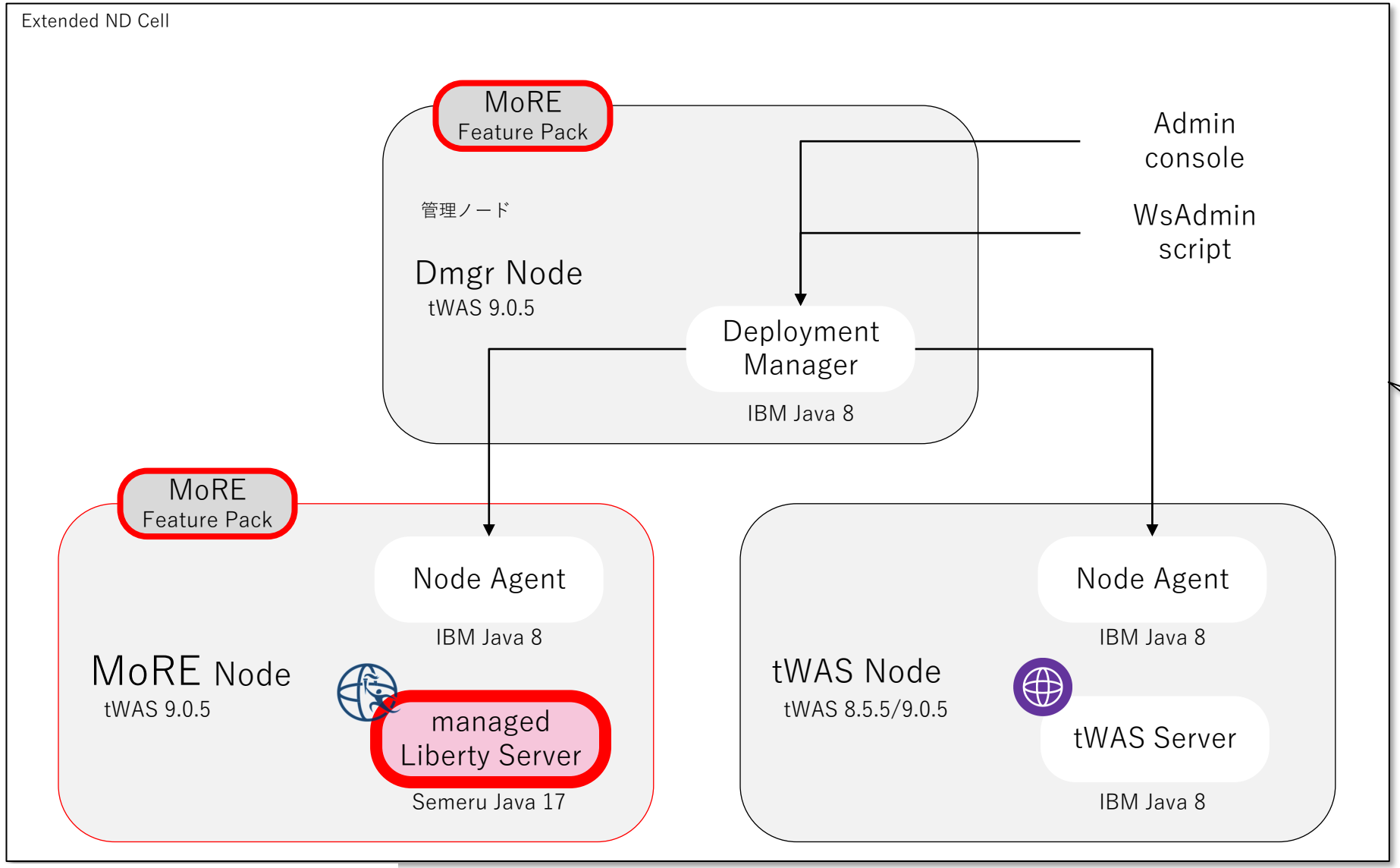


Java 8: Business risk you can't ignore (webinar)

ともかく延命だけしたい場合
(システム統合が遅れているなどの理由)

並行稼働させる場合は２つ目だけ提案。
最新JavaSEで開発は３つ目も合わせて提案。

Modernized Runtime Extension for Java (MoRE)



EARで使用可能な
追加オファリング

従来型WAS NDのCell構成にLibertyを組み込み管理を維持しつつ、Java 17のアプリを実行

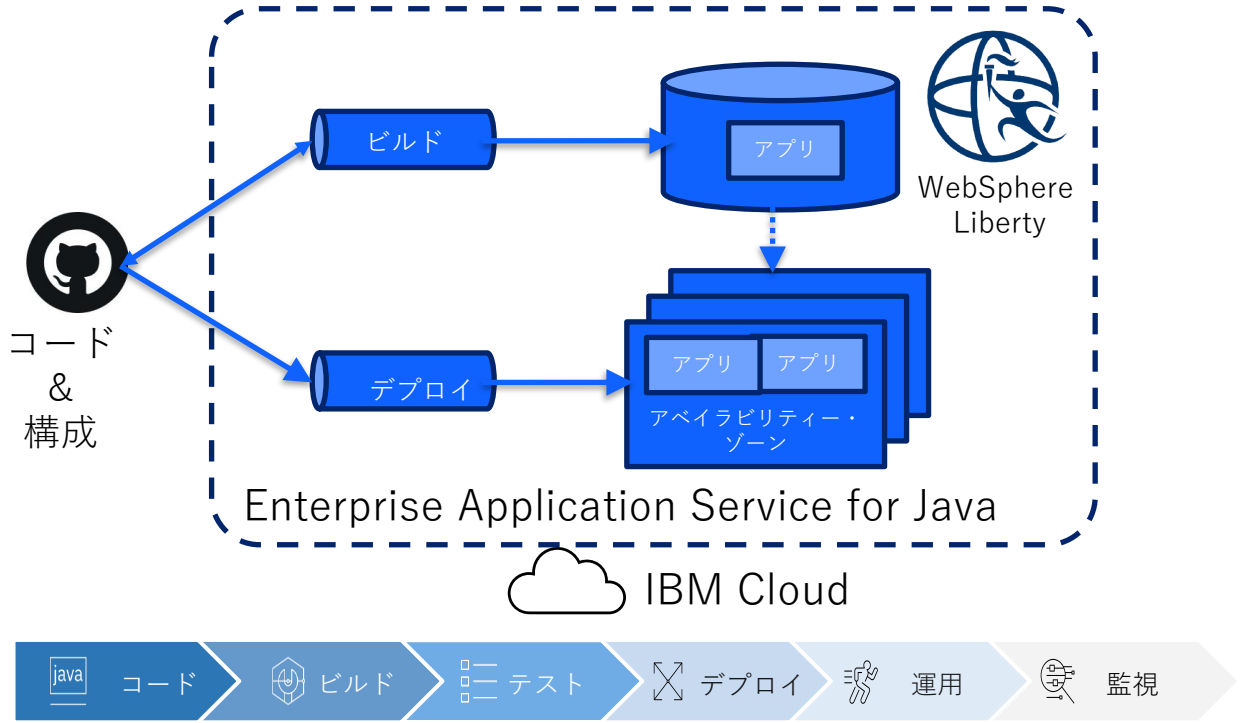
並行稼働、あるいは運用維持したい。
Java17へのバージョンアップや、Liberty化を推進。

２つ目

Enterprise Application Service for Java (EASeJ)

■ アプリケーションのモダナイゼーションのために最適化された**フルマネージドのサービス**

– 軽量で効率的，クラウドに最適化された**Libertyランタイムを PaaSで提供**
– ワークロードのリクエストに基づいて動的に拡張するマネージド型のランタイム
– ビルドとデプロイの**パイプラインを構築済み**
– CI/CD，高可用性，災害対策，セキュリティーのベストプラクティスを組み込み
– オンプレミス環境からPaaSへの移行ツール (**Application Modernization Accelerator**) も提供



Revisit

Q1. 現在お使いのWebSphereランタイムは？
A1. **従来**のWebSphereランタイム
A2. WebSphere **Liberty**ランタイム
A3. どちらも利用

A2 / A3 → ヒアリング終了・通常の質問
「現在お困りのことはありますか？」
「不明点があればTechを呼びます」etc.

A1
確認方法：
管理コンソールあり->従来型
管理コンソールなし(server.xmlで管理)->Liberty

Q2. お客様自身がアプリケーションを作成

Yes → Libertyへの移行をお薦めしてください
「モダナイゼーションによりコスト削減
運用の高品質化，開発の迅速化が可能」

No
ソリューションベンダー提供の
あぷりを使用している場合など

Q3.ソリューションベンダーとのコンタクトが可能？

No → EARによる塩漬けを提案

Yes
ベンダーのコンタクト先が入手でき，連絡がついた

IBMのTech Salesからベンダーにliberty化を依頼しますのでお知らせください

BP様パターンについて

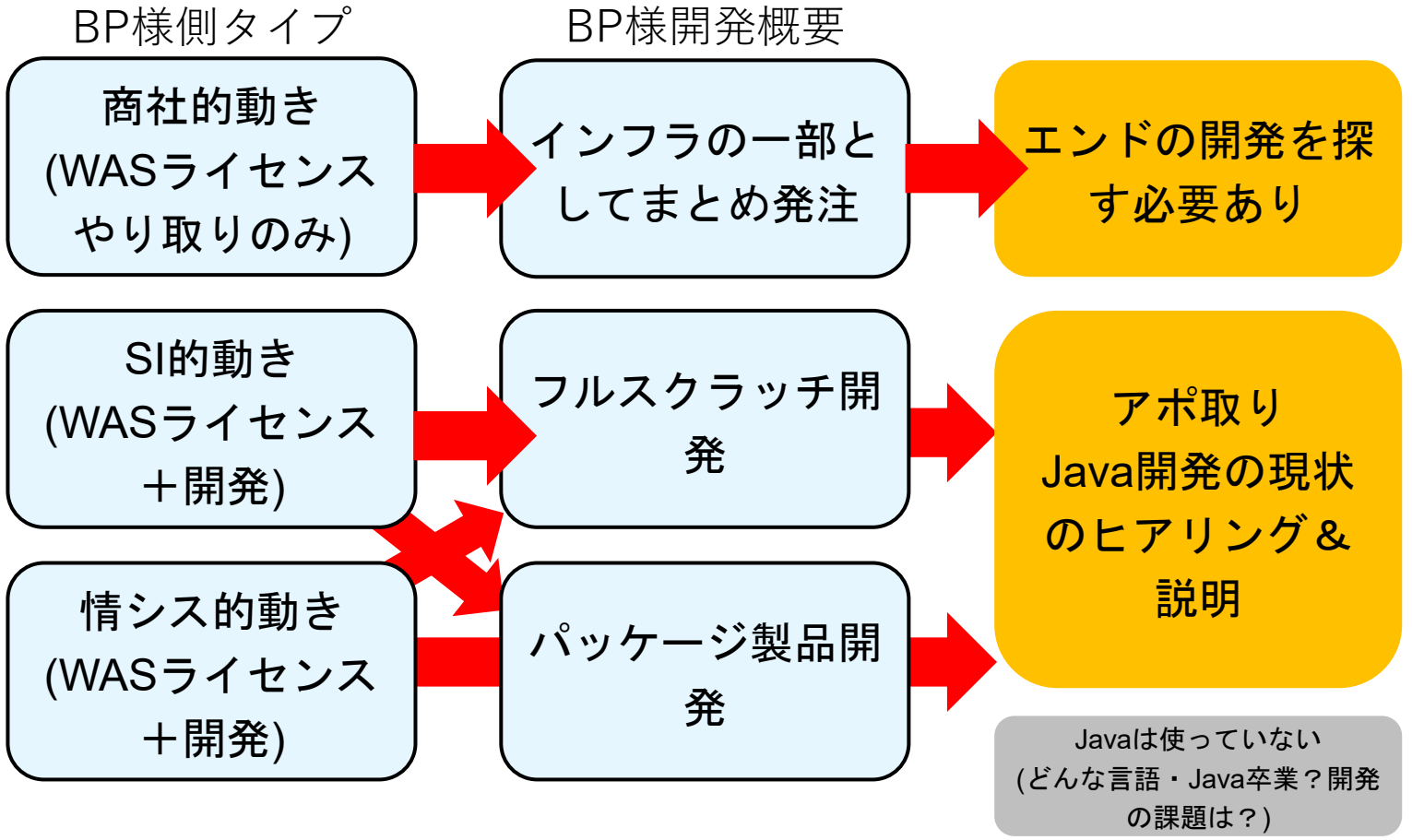
Revisit

- IT部門が独立してシステム開発する。
- 本社要件に応じて開発、その簿パッケージ開発で生業となり独立成長している場合あり。

ライセンスのやり取りのみで、開発していないケース。
商材知識や用語、Javaの背景などBPへ知識をINPUT、説明へ行っていただく。

開発しているためJavaもWASもよくわかっているケース。
JavaやWAS、フレームワークやJava開発状況などをお聞きする。

IT部門が独立してシステム開発する。
本社要件に応じて開発、その簿パッケージ開発で生業となり独立成長している場合あり。JavaやWAS、フレームワークや開発状況などをお聞きする。



他にどんなパターンがあるか？

再掲）Revisitにおける目的

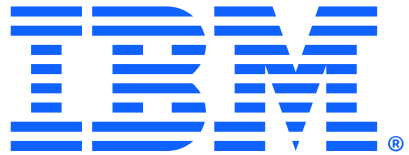
1. Java8サポート終了に対して、既存のWAS利用の維持とアップセル
   - WAS継続利用のためのLiberty化移行推進
   - クラウドファースト企業への訴求
   - 開発・検証環境利用としての訴求

   →最新Liberty ライセンス(Jsphere)への移行(買い替え)

2. お客様のご利用環境・課題を把握することで新たなビジネス機会を創出
   - アプリケーションモダナイズ → WCAの生成AIを利用した移行や開発サイクルの効率化
   - 運用の高度化・自動化 → Instana/Trubonomicの提案に繋げる

3. 既存のWASでないJava環境をお持ちのお客様の獲得
   - Java8の継続利用が必要なお客様へのアプローチ
   - Java8からバージョンアップを低コストにしたいお客様へのアプローチ

   →WASへのWin back、最新Libertyライセンス(Jsphere)やWCAにて既存Java8環境の最適化



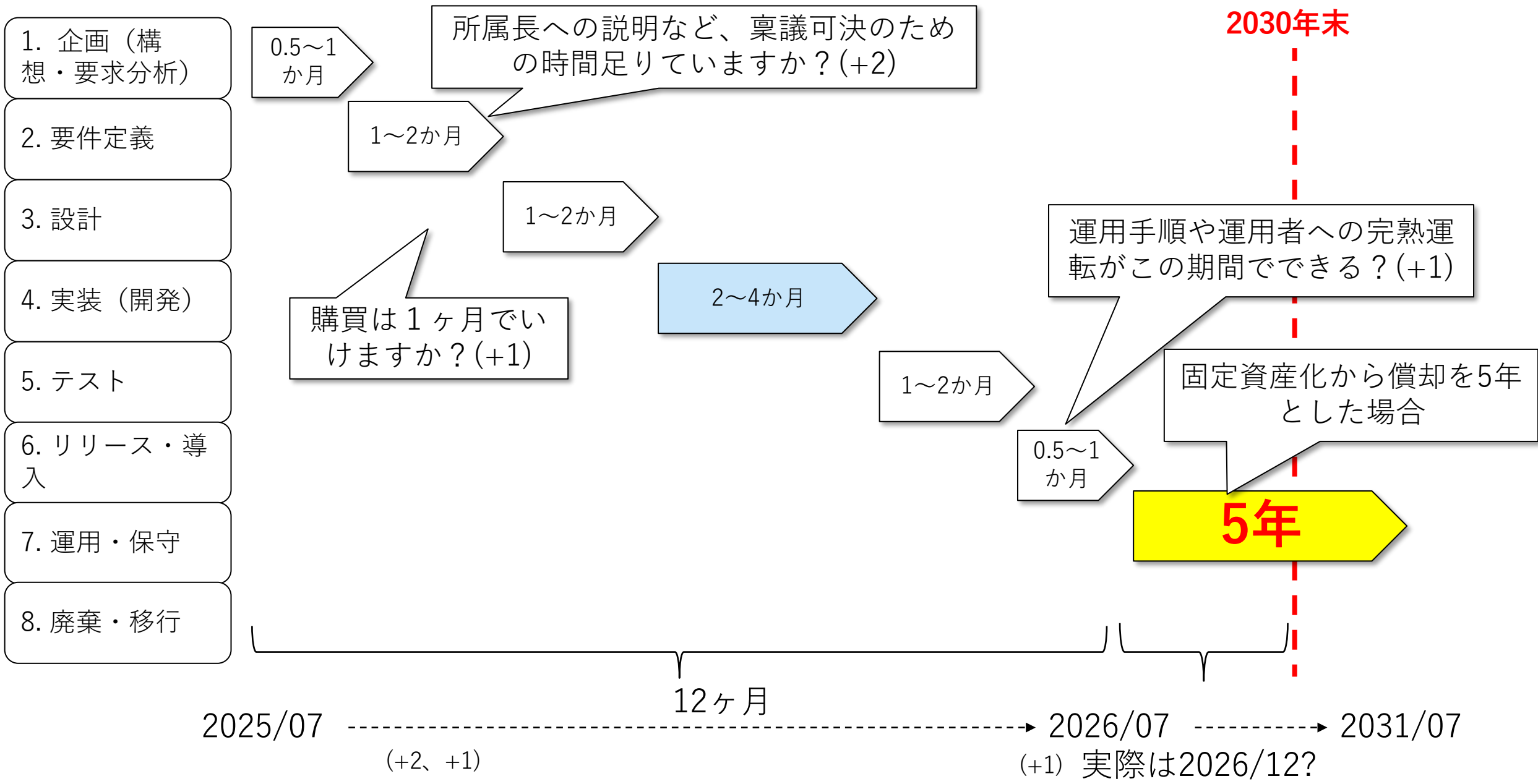
「まだ5年ある」問題

開発工程について

■ ソフトウエアもライフサイクルがあります。以下はライサイクルの最低項目の例。

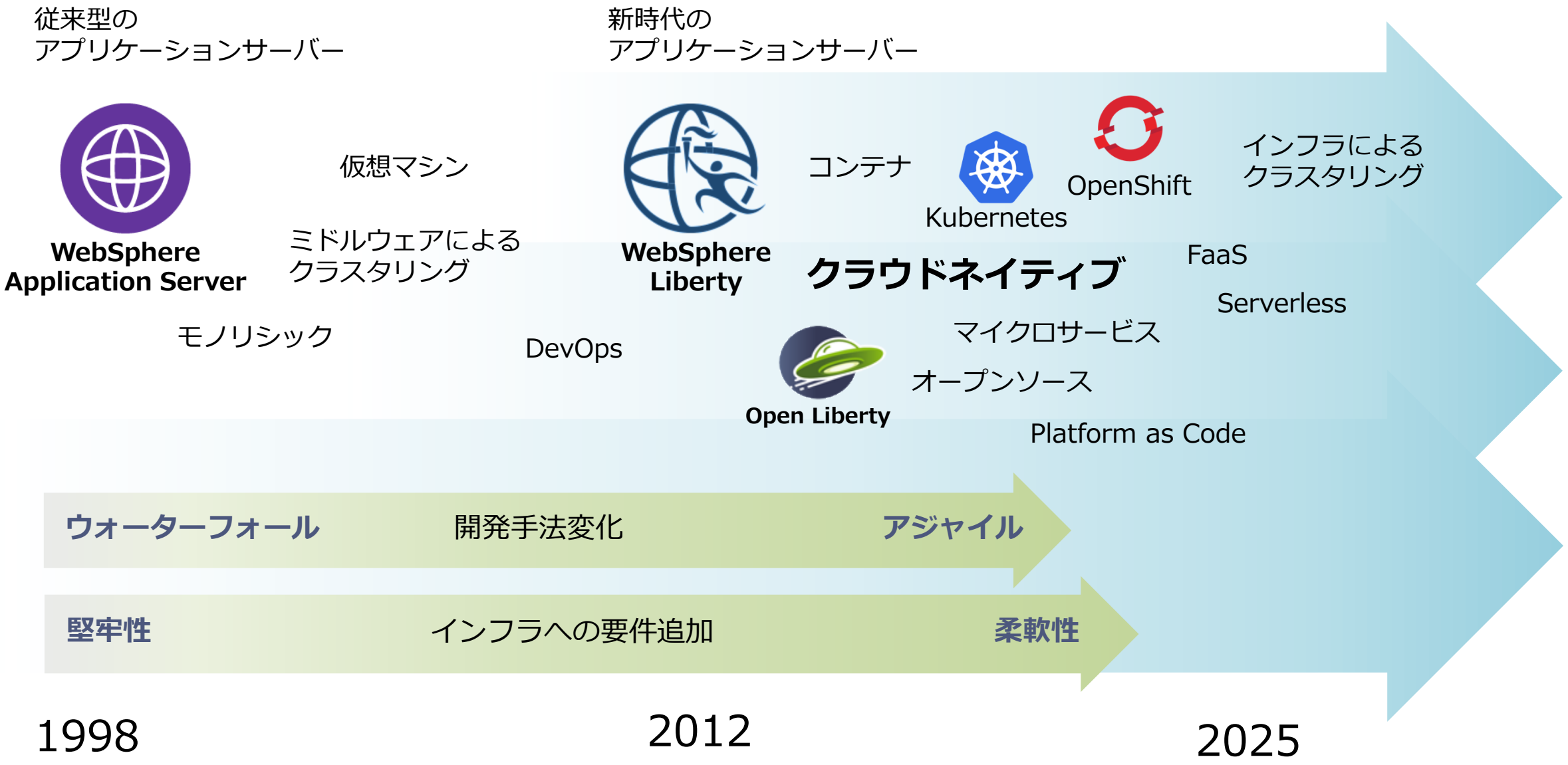
1. 📝 企画（構想・要求分析）	- ビジネス課題の明確化- ユーザー調査・KPI設計	0.5〜1か月
2. 🔍 要件定義	- ユーザー要求の取りまとめ- システム要件の明文化（機能／非機能）	1〜2か月
3. 📐 設計	- 基本設計：画面構成・機能一覧- 詳細設計：DB設計・API仕様	1〜2か月
4. 💻 実装（開発）	- コーディング- 単体テスト	2〜4か月（並行してテスト準備）
5. 🧪 テスト	- 結合テスト、システムテスト- 受け入れテスト（UAT）	1〜2か月
6. 🚀 リリース・導入	- 本番リリース準備- ユーザー教育・移行手続き	0.5〜1か月
7. 🛠️ 運用・保守	- 障害対応- 軽微な改善対応	継続（数年）（月次の更新サイクルなど）
8. 🧹 廃棄・移行	- サービス終了通知- データ削除・移行支援	0.5〜1か月（終了時点で1回）

ソフトウエア開発でのライサイクル：開発にかかる時間



WASが注目される理由

WebSphereをとりまくIT技術の進化



そもそも「WebSphere」って、何をする製品？

Java言語を使い、企業向け機能を備え、開発される
「業務アプリケーション」を
実行するためのプラットフォーム

業務アプリケーションの例

- ブラウザから利用するWebアプリケーション
- 他のシステムからHTTPで接続して利用するAPI

既製品のソフトウェア/SaaS では対応できない業務に対し、お客様自身で、独自の処理を実装し、目的の業務を実現する場合に使用される。

業務アプリケーションの実装に使われるプログラミング言語

業務アプリケーションの実装に使われるプログラミング言語

WebSphere

特に、日本の業務システムで、
最も広く使われているプログラミング言語

Java8に関するサポート終了状況

Java8仕様を活用している製品/OSSのサポート終了が増加している。

Java8 EoSに向けての各社対応状況

JBoss EAP8（最新版）はJava8は対応していない。

*お客様はEAP7使っている？

Oracle WebLogic も最新版はJava8は対応していない。

Jakartaは未対応

Fujitsu Enterprise Application Platform :Liberty

Fujitsu InterStage :Traditional

IBM Liberty 26.0.X代は2026年9月リリース版でサポート終了。

IBM Tranditional WASも2030年でサポート終了。

最新技術の採用ができない：利用者のための実装要望にお応えすることが難しい。

技術負債の維持：古い技術者を抱え続けなくてはいけない。

市場・技術周辺状況

エンジン以外の周辺ライブラリやフレームワーク自体もJava8 Supportが終了となっている。

技術者が少なくなってきている。

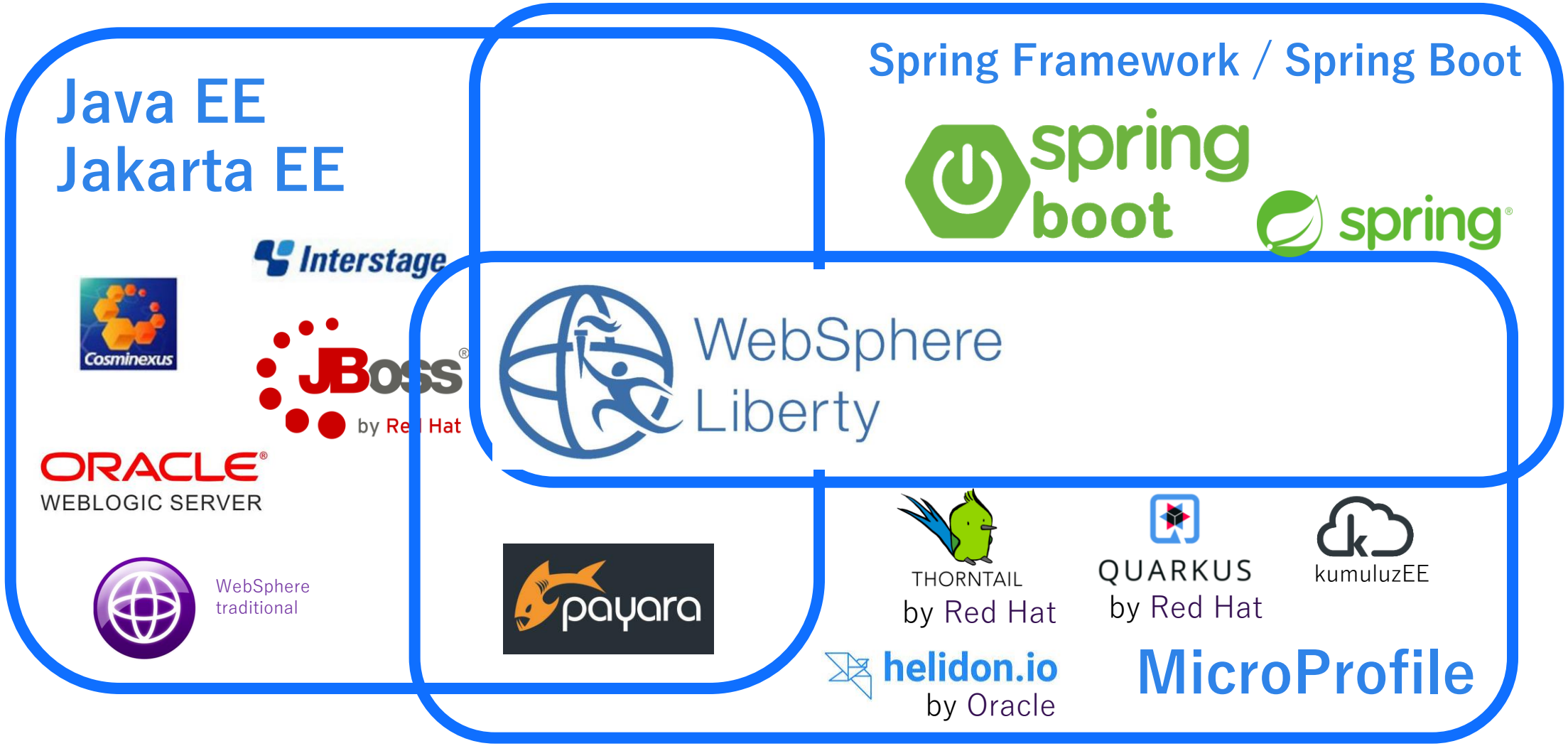
Libertyとtraditional：用途に応じたランタイムの選択

- **WAS traditional**（WebSphere traditional / tWAS / 従来型WAS）
  - **既存資産の活用**を目的としたランタイム
    - 旧WASでおこなっていた**従来の運用を継続**したいお客様
  - Libertyで対応していない**旧API**を使用しているアプリケーションの実行環境として
    - JAX-RPCやEntity Beanなど
  - **今後，新機能の実装や新しい仕様への対応は行われない**
    - Java EE 7 / Java 8に対応した2016年出荷のV9.0が**最後のバージョン**

- **WAS Liberty** (WebSphere Liberty)
  - モダンなアプリケーション開発・サーバー運用に対応した**新時代のランタイム**
    - 軽量さを活かしたAgile開発やCD（継続的デリバリー）
    - ツールによる運用の自動化・DevOps（Platform as a Code / Immutable Infrastructure）
  - WAS traditionalとは，運用に対する設計思想が根本的に異なる
  - **クラウド**での使用やコンテナ環境，リソースの限定されたIoT環境にも最適
  - 今後も**継続して進化**を続ける

Java 8や最新Java EEだけでなく、MicroProfileにも対応しているWebSphere Libertyが、移行先としてお勧めです

エンタープライズJavaの業界動向



各社のモダナイズへの取り組み

開発者のスキルや既存アプリを活用したままクラウドネイティブへの対応ができるのはIBMのWebSphereだけ

従来型ランタイム

JBoss by Red Hat
JAKARTA EE
Java EE
MICROPROFILE
*1

ORACLE WEBLOGIC SERVER
Java EE

WebSphere traditional
Java EE

クラウドネイティブ対応・次世代ランタイム

QUARKUS by Red Hat
MICROPROFILE

helidon.io by Oracle
MICROPROFILE

WebSphere Liberty by IBM
MICROPROFILE
Java EE
JAKARTA EE

*1 限定的にサポート

プログラミング言語Javaの特長

■“Write once、 Run anywhere.”
- –Java仮想マシン（JVM）用の中間コードにコンパイルされる
- –JVMが移植された環境であれば、同じプログラムが同じように稼働する
  - • Windowsで開発→本番はLinux、Macで開発→本番はzOSなどが普通にできる

金融系で導入実績長い

■強く型付けされた静的言語
- –変数の型が実行時ではなくプログラムコード記述時に決定されている
- –自由度が少ない ←→ 人の書いたコードが読みやすい
  - • 多人数で開発するプログラムや、広く再利用されるプログラムの開発に適している

■後方互換性を重視する文化
- –20年前のプログラムが、昨今の環境で普通に動く
- –それまで動いていたコードが動かなくなる「破壊的な変更」に強く抵抗するコミュニティ

エンタープライズJavaの概要

■Java EE / Jakarta EE

– 2000年ごろからバージョンアップを続けている業界標準仕様
– 「アプリケーションサーバー」という実装形態が
昨今のクラウドネイティブ環境と親和性がないと**敬遠されがち**

■Spring Framework / Spring Boot

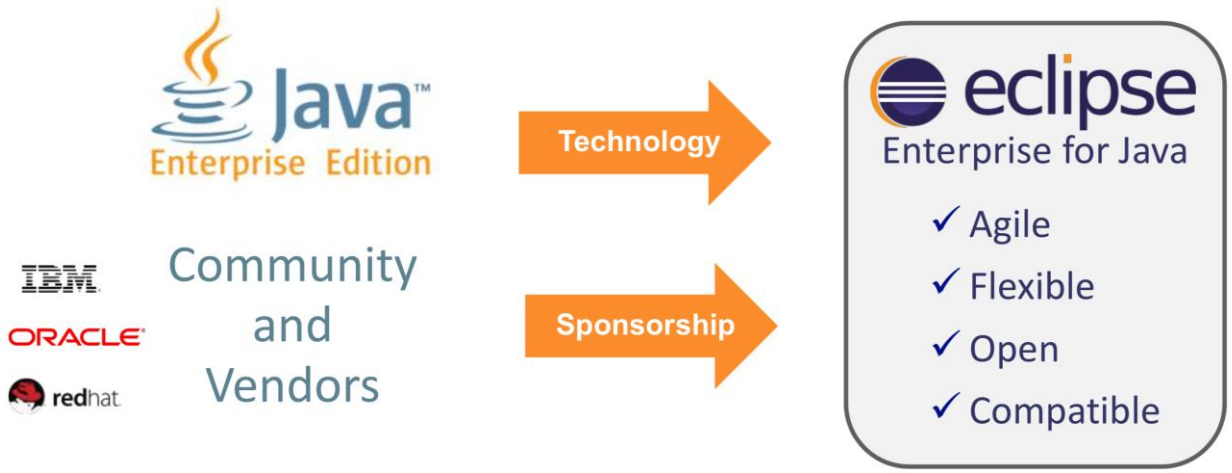
– 旧Pivotal社（現在はVMwareに買収）によるフレームワーク
– 先進的で高機能、DevOpsがやりやすく急速に**人気が高まっている**
– バージョンアップへの追随が大変

■MicroProfile

– マイクロサービス・アーキテクチャーでアプリを実装するための新しい標準仕様
– 2016年から提供され、**注目を集めている**

Java EE / Jakarta EE

- Java EE （Java Enterprise Edition）
  - Oracle主導のJCP（Java Community Process）で策定されていたエンタープライズJavaの標準仕様
  - 2015年のJava EE 8が最後の仕様となった
  - 2016年より、OracleのJava仕様にたいする活動低下により、更新が停滞
- Jakarta EE（Eclipse）
  - Eclipse Foundationに移管され、コミュニティ主導で策定されるエンタープライズJavaの仕様
  - IBM / Oracle / Red Hat / Fujitsu など、主要なJava関連ベンダーの多くが参画
  - 2019年：Jakarta EE 8　Java EE 8と同等の仕様 / Jakarta EEによる策定・公開プロセスの元での再定義
  - 2020年：Jakarta EE 9　商標上の問題がある**javax名前空間からjakarta名前空間への移行**
  - 2021年：Jakarta EE 9.1 Java SE 11に対応
  - 2022年：Jakarta EE 10　新しい機能を追加した新バージョン



MicroProfile

■ Javaでマイクロサービス・アーキテクチャーを採用して
アプリを実装する際に必要な機能を複数ベンダーで標準化

– Config　分散環境で構成情報を共有
– Health Check　サービスの稼働状況をチェック
– Fault Tolerance　サービス呼出しのエラーハンドリングやリトライを自動化
他Open Tracing、Metrics、OpenAPI、TypeSafe REST clientなど

■ 非常に進歩が速いのが特徴

2016/9	MicroProfile 1.0	2018/6	MicroProfile 2.0	2019/6	MicroProfile 3.0
2017/8	MicroProfile 1.1	2018/10	MicroProfile 2.1	2019/10	MicroProfile 3.1
2017/9	MicroProfile 1.2	2019/2	MicroProfile 2.2	2019/11	MicroProfile 3.2
2018/1	MicroProfile 1.3			2020/2	MicroProfile 3.3
2018/6	MicroProfile 1.4				
2020/12	MicroProfile 4.0	2022/1	MicroProfile 5.0	2024/12	MicroProfile 7.0
2021/8	MicroProfile 4.1	2023/1	MicroProfile 6.0		
		2023/10	MicroProfile 6.1		

OSSのTomcatの例

名前空間の変更

Tomcat	Servlet 3.0	Servlet 3.1	Servlet 4.0	Servlet 5.0
Tomcat Version 7.0	○			
Tomcat Version 8.0 / 8.5	×	○		
Tomcat Version 9.0	×	×	○	
Tomcat Version 10.0	×	×	×	○

古いものは順次廃止

名前空間の変更

Liberty	Servlet 3.0	Servlet 3.1	Servlet 4.0	Servlet 5.0
Liberty V8.5～	○			
Liberty V8.5.5～	○	○		
Liberty V8.5.5.6～	○	○	○	
Liberty V20.0.0.12 Beta～	○	○	○	○

古いものも引き続き対応

Tomcat v10.0必須になった場合、Servlet5.0と名前空間変更、つまりコード変更も必須となる。

Libertyが最新版でも、Servlet3.0や4.0を使用できる。

移行のためのコード改修費用が「かなり」抑えられます

提供されている二つのランタイム

バージョン	提供されるWASランタイム	
2011年7月 **WAS V8.0**		WAS **Java EE6** 完全対応
2012年7月 **WAS V8.5**	WAS Libertyプロファイル Servlet/JSPなど基本機能	WAS Fullプロファイル **Java EE6** 完全対応
2013年6月 **WAS V8.5.5**	WAS Libertyプロファイル **Java EE6 Web Profile**対応	WAS Fullプロファイル **Java EE6** 完全対応
2015年6月 **WAS V8.5.5.6**	WAS Libertyプロファイル **Java EE7** 完全対応	WAS Fullプロファイル **Java EE6** 完全対応
2016年6月 **WAS V9.0**	**WAS Liberty** **Java EE7/8** 完全対応	**WAS traditional** **Java EE7** 完全対応

最初のLibertyランタイムは
機能が限定されていたので、
従来型のランタイムは
全機能を提供しているということで
Fullプロファイルと呼ばれた

V8.5.xの途中から逆転し
V9.0ではLibertyランタイムの方が
より多くの技術に対応している

Libertyとtraditional：用途に応じたランタイムの選択

- **WAS traditional**（WebSphere traditional / tWAS）
  - **既存資産の活用**を目的としたランタイム
    - 旧WASでおこなっていた**従来の運用を継続**したいお客様
  - WAS Libertyで対応していない**旧API**を使用しているアプリケーションの実行環境として
    - JAX-RPCやEntity Beanなど
  - 今後、新機能の実装や新しい仕様への対応は行われない
    - Java EE 7 / Java 8に対応した2016年出荷のV9.0が**最後のバージョン**

- **WAS Liberty** (WebSphere Liberty)
  - モダンなアプリケーション開発・サーバー運用に対応した**新時代のランタイム**
    - 軽量さを活かしたAgile開発やCD（継続的デリバリー）
    - ツールによる運用の自動化・DevOps（Platform as a Code / Immutable Infrastructure）
  - WAS traditionalとは、運用に対する設計思想が根本的に異なる
  - **クラウド**での使用やコンテナ環境、リソースの限定されたIoT環境にも最適
  - 今後も**継続して進化**を続ける

Libertyランタイムとtraditionalランタイムで対応している仕様

WebSphere traditional

Java EE 6 (WAS v8.5.5.x)
Java EE 7 (WAS v9.0.5.x)

Java SE 8

WebSphere Liberty
Open Liberty

Java EE 6
Java EE 7
Java EE 8
Jakarta EE 9.1
Jakarta EE 10

Java SE 8
Java SE 11
Java SE 17
Java SE 21
Java SE 24

MicroProfile 1.0
MicroProfile 1.2
...
MicroProfile 4.0
MicroProfile 4.1
MicroProfile 5.0
MicroProfile 6.0
MicroProfile 6.1
MicroProfile 7.0

2025年5月現在

WAS Traditionalランタイムについてよくある質問

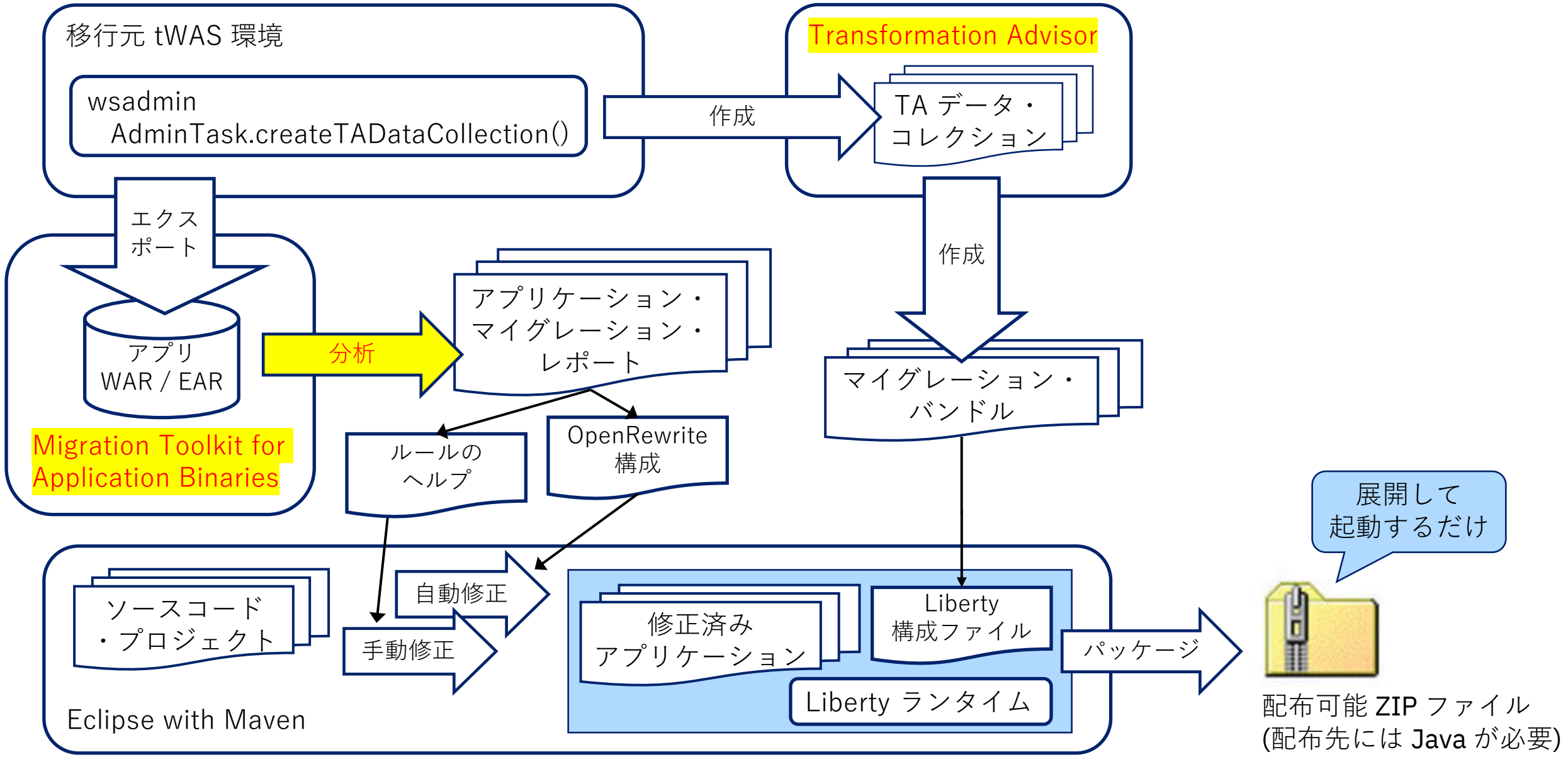
- V9.0の次のバージョンは出ないの？
  - でません。WAS Traditionalランタイムは、V9.0.xが最後のバージョンです。
- V9.0は、Java 11対応Java 17対応はされるの？
  - されません。今後、WAS Traditionalランタイムは、新しい仕様やプラットフォームへの対応はおこなわれません。
- 2030年までは新規導入して利用できるの？
  - 可能ですが、お薦めしません。WAS V8.5 Fullプロファイル / WAS V9.0 Traditionalランタイムは2030年までのサポートを表明していますが、提供されている機能の陳腐化が始まっています。システムのライフサイクルをとおして必要な機能を提供できないリスクがあります。
- WAS Traditionalランタイムをクラウド上で利用することはできるの？
  - 後述のライセンスの要件がクリアできるのであれば可能です。
    ただし、ホスト名やIPアドレスなどが固定されたIaaS環境を利用ください。
- WAS Traditionalランタイムをコンテナ上で利用することはできるの？
  - 可能ですが、お薦めしません。
    GUIによる管理ができない、ND版が利用できない、Sessionの共有ができないなど、多くの制限があります。

2030年よりも前にサポートが終了するコンポーネント

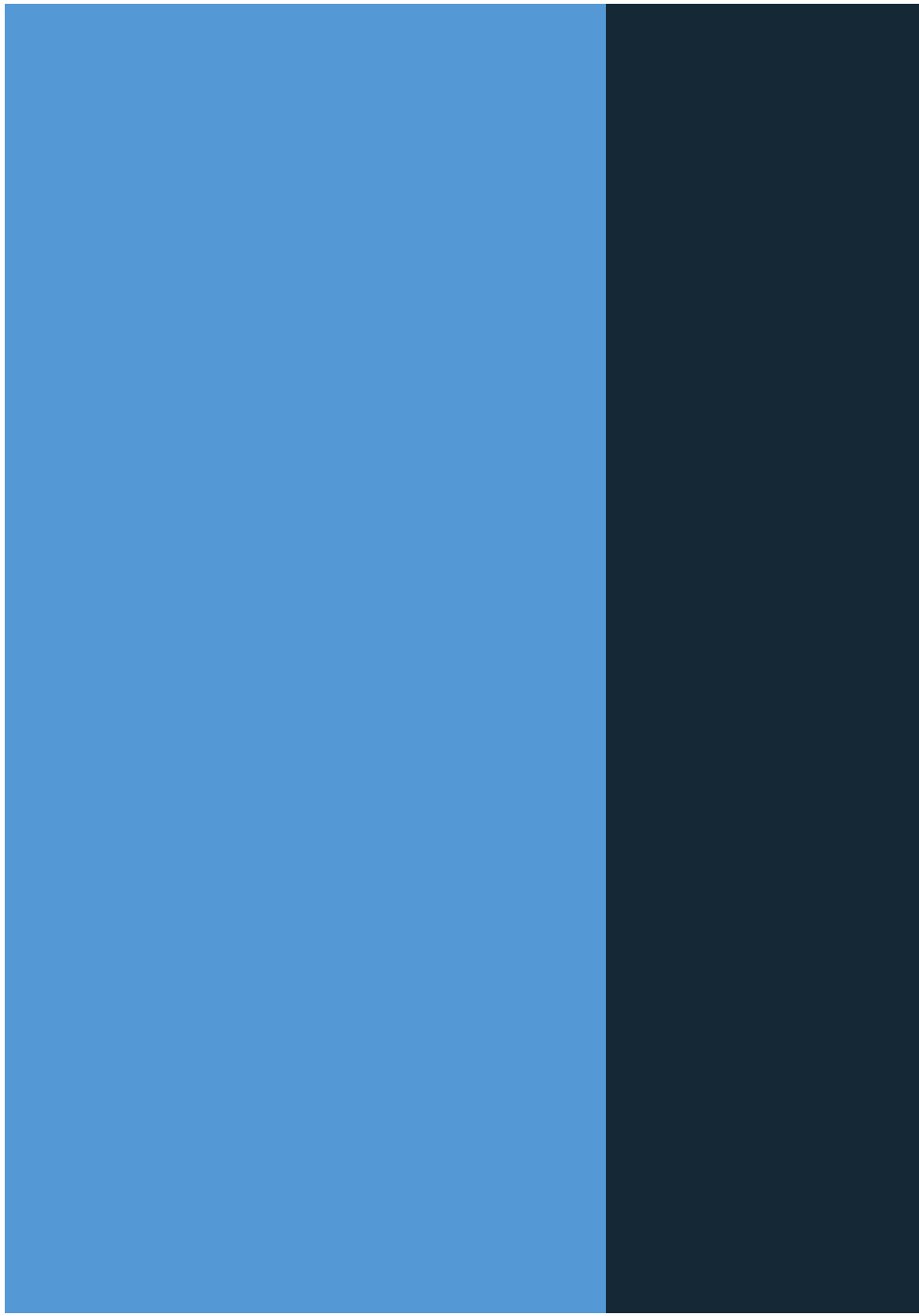
WAS V8.5 Full Profile/9.0 traditional本体は少なくとも2030年までサポートされますが、
構成するサブコンポーネントなどには、それより前にサポートが終了するものがあるので注意

- 2018年4月：WAS V8.5のJava 6サポート終了
- 2019年9月：Edge ComponentのLoad Balancer for IPv4 (LLB)のサポート終了
- 2022年7月：WAS V8.5のJava 7サポート終了
- 2022年9月：SolarisならびにHP-UXプラットフォームのサポート終了
- 2024年9月：IBM HTTP Server/WebSphere Pluginの32bit版のサポート終了
- 2024年9月：Edge ComponentのCaching Proxyのサポート終了(ULBはtWASに追従)
- 2025年12月： IBM HTTP Server V8.5のサポート終子(tWASに追従)

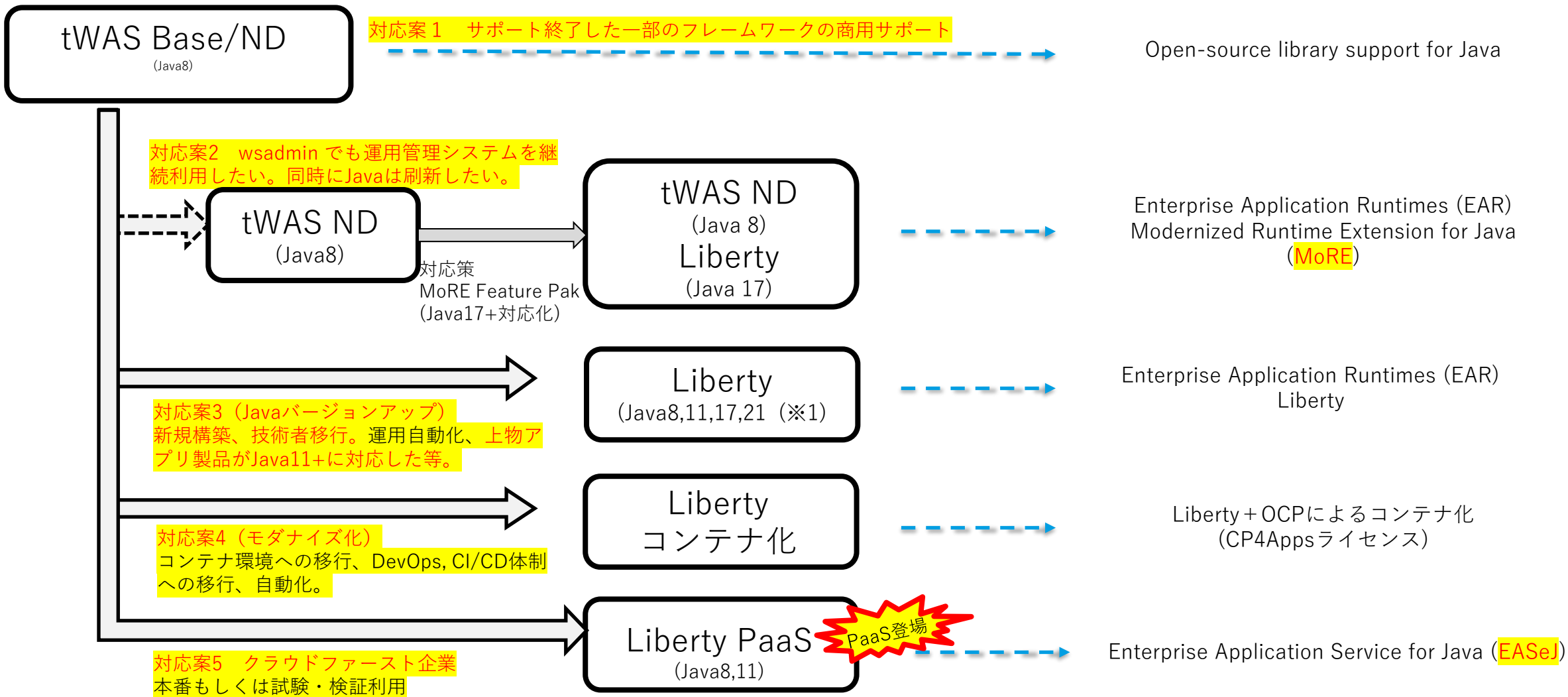
WebSphere Libertyへのマイグレーション・ステップ



Java8問題を支えるオファーリング



tWASからWAS Libertyへの移行について



※1:旧AdoptOpenJDKコミュニティー（現Eclipse Adoptium）およびIBMが提供する、HotSpotベースの Eclipse TemurinとOpenJ9ベースのIBM Semeru Runtimesに対応しています。IBM Runtimes for Businessは、これらのOpenJDKバイナリ（Eclipse TemurinおよびIBM Semeru Runtimes）を包括的に商用サポートし、OpenJDK 8、11、17、21のLTSバージョンに対応しています。さらに、Javaアプリケーションの監視および管理機能も提供しています。

Java 戦略のための費用対効果の高いオプション

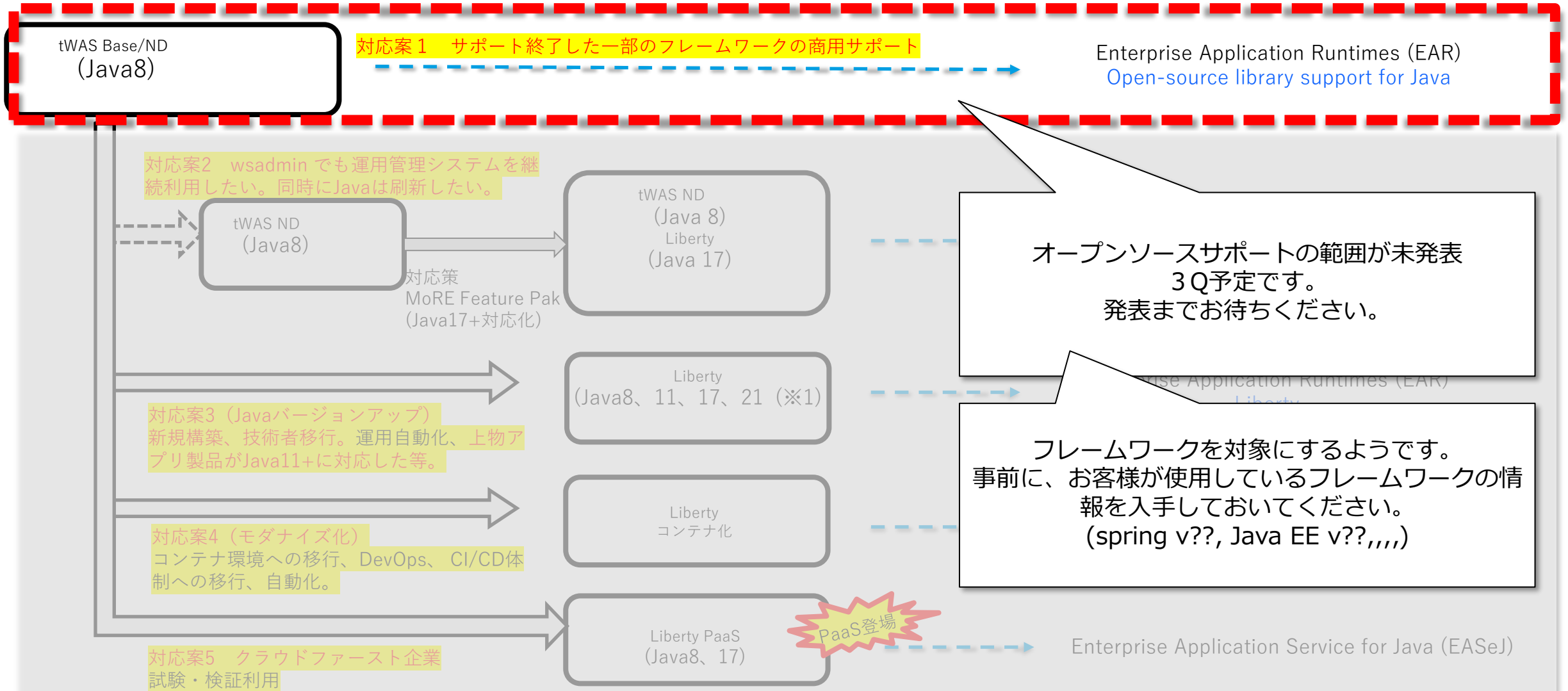
	IBM JSphere Suite for Java					
バンドル	Enterprise Application Runtimes (EAR)			Enterprise Application Service for Java (EASeJ)	Cloud Pak for Applications (CP4Apps)	代替ソリューション (限定的な自動化)
コンポーネント	Open-Source Library Support for Java	Modernized Runtime Extension for Java (MoRE)	Liberty		Liberty	
モダナイゼーション戦略 →	Java 8 を維持	Java をアップグレード (WAS Admin)	クラウドに移行 + Java をアップグレード + コンテナを導入 + マイクロサービス	IBM マネージド・クラウドに移行 + Java をアップグレード	ハイブリッドクラウドに移行 + Java をアップグレード + コンテナを導入 + マイクロサービス	各種
時間の節約*	100%	95%	65%	80%	65%	なし
移行から得られる変革的なメリット	該当なし	中	高	高	高	高

3Q 中発表予定

自社資産のそれぞれのアプリケーションにとって合理的な戦略とオファリングを使用

* IBM Consulting のモダナイゼーション・エンゲージメント分析と、支援なしの 700 時間に基づく (アプリケーションあたり)

tWASからWAS Libertyへの移行について



※1:旧AdoptOpenJDKコミュニティー（現Eclipse Adoptium）およびIBMが提供する、HotSpotベースのEclipse TemurinとOpenJ9ベースのIBM Semeru Runtimesに対応しています。IBM Runtimes for Businessは、これらのOpenJDKバイナリ（Eclipse TemurinおよびIBM Semeru Runtimes）を包括的に商用サポートし、OpenJDK 8、11、17、21のLTSバージョンに対応しています。さらに、Javaアプリケーションの監視および管理機能も提供しています。

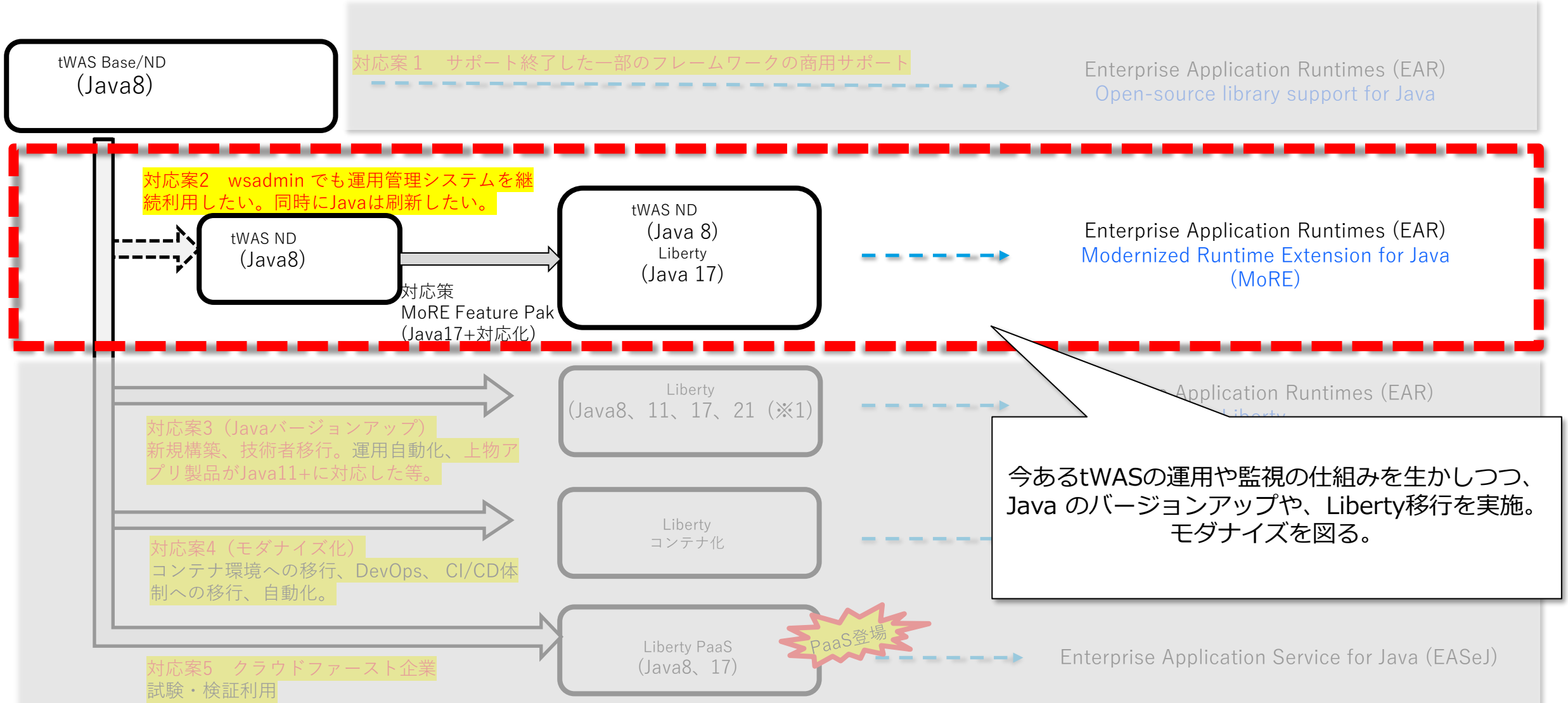
Java 戦略のための費用対効果の高いオプション

	IBM JSphere Suite for Java					
バンドル	Enterprise Application Runtimes (EAR)			Enterprise Application Service for Java (EASeJ)	Cloud Pak for Applications (CP4Apps)	代替ソリューション (限定的な自動化)
コンポーネント	Open-Source Library Support for Java	Modernized Runtime Extension for Java (MoRE)	Liberty		Liberty	
モダナイゼーション戦略 →	Java 8 を維持	Java をアップグレード (WAS Admin)	クラウドに移行 + Java をアップグレード + コンテナを導入 + マイクロサービス	IBM マネージド・クラウドに移行 + Java をアップグレード	ハイブリッドクラウドに移行 + Java をアップグレード + コンテナを導入 + マイクロサービス	各種
時間の節約*	100%	95%	65%	80%	65%	なし
移行から得られる変革的なメリット	3Q 中発表予定	中	高	高	高	高

自社資産のそれぞれのアプリケーションにとって合理的な戦略とオファリングを使用

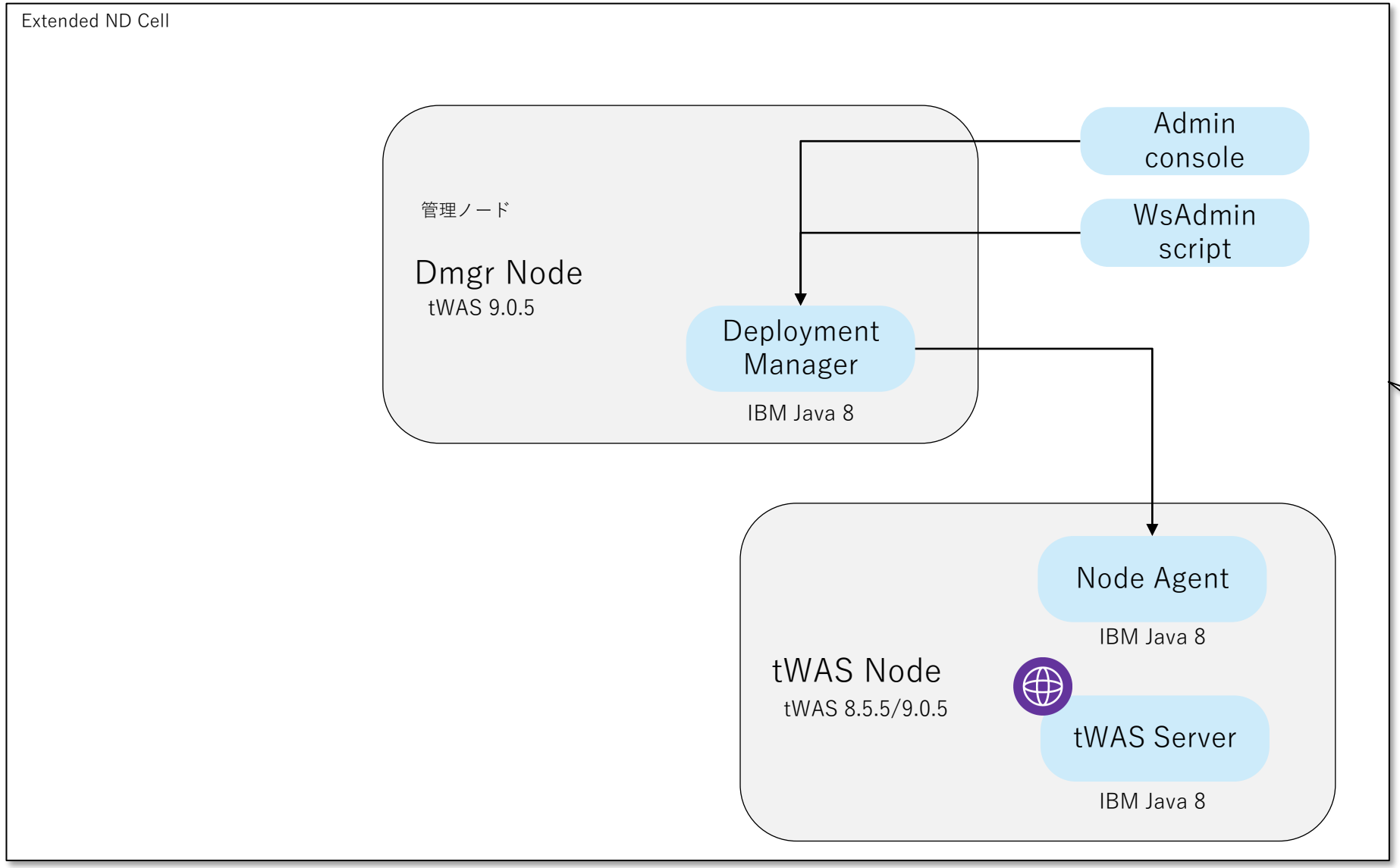
* IBM Consulting のモダナイゼーション・エンゲージメント分析と、支援なしの 700 時間に基づく (アプリケーションあたり)

tWASからWAS Libertyへの移行について



※1:旧AdoptOpenJDKコミュニティー（現Eclipse Adoptium）およびIBMが提供する、HotSpotベースのEclipse TemurinとOpenJ9ベースのIBM Semeru Runtimesに対応しています。IBM Runtimes for Businessは、これらのOpenJDKバイナリ（Eclipse TemurinおよびIBM Semeru Runtimes）を包括的に商用サポートし、OpenJDK 8、11、17、21のLTSバージョンに対応しています。さらに、Javaアプリケーションの監視および管理機能も提供しています。

Modernized Runtime Extension for Java (MoRE)

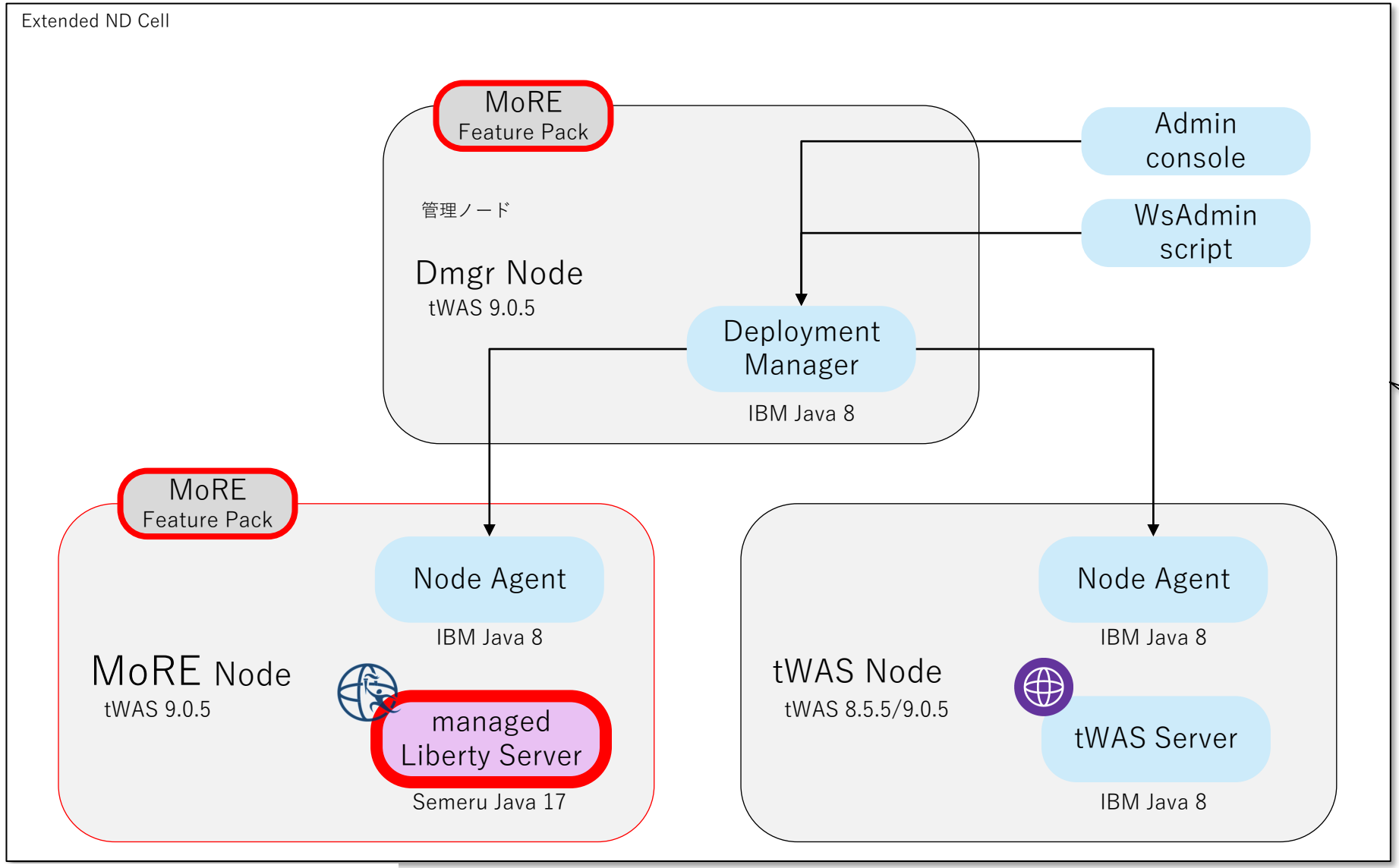


EARで使用可能な
追加オファリング

従来型WAS NDのCell構成にLibertyを組み込み管理を維持しつつ、Java 17のアプリを実行

Java17へのバージョンアップや、
Liberty化を推進

Modernized Runtime Extension for Java (MoRE)



EARで使用可能な
追加オファリング

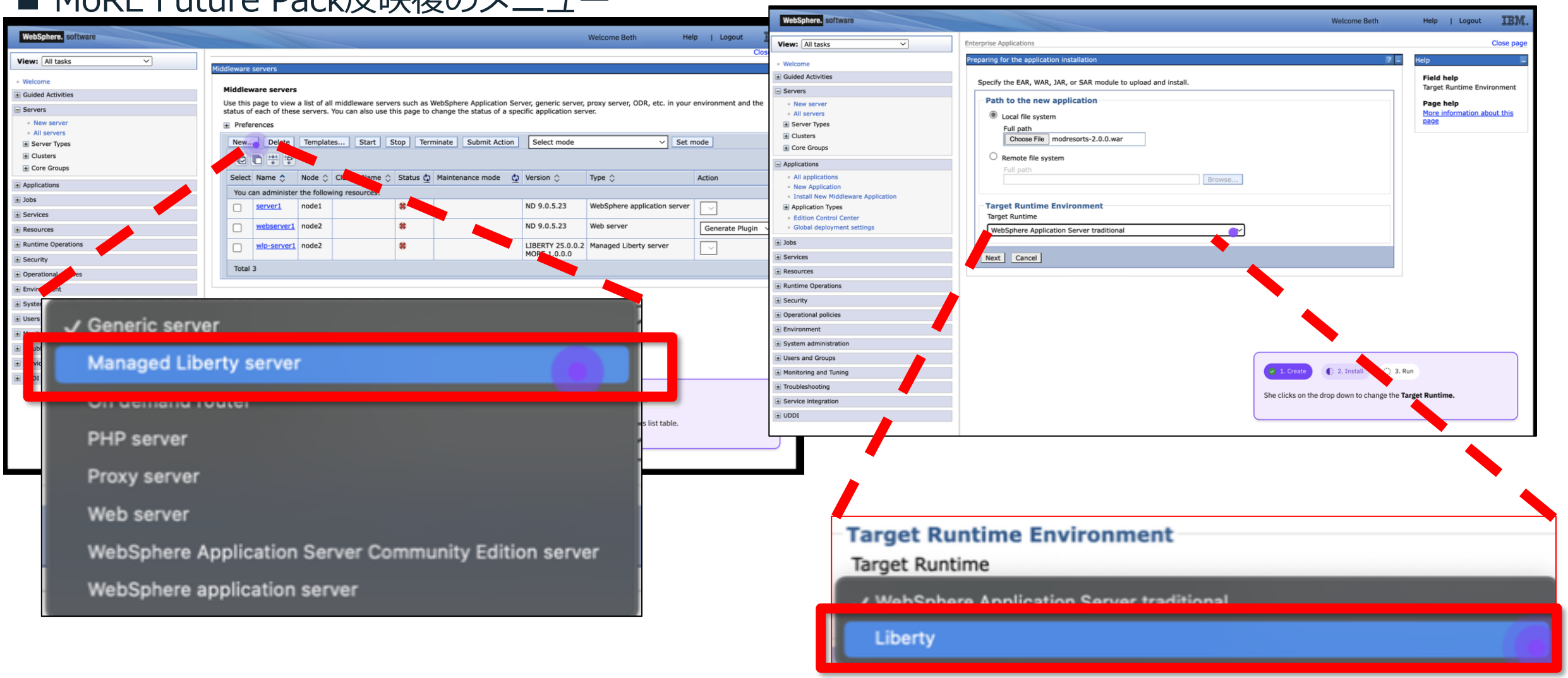
従来型WAS NDのCell構成にLibertyを組み込み管理を
維持しつつ、Java 17のアプリを実行

Java17へのバージョンアップや、
Liberty化を推進

IBM Application Development - 2025

Modernized Runtime Extension for Java (MoRE)

■ MoRE Future Pack反映後のメニュー



コードの変更がハードル問題

■ コードの変更がハードルという場合

– 今回発表した「Enterprise Application Runtimes (EAR)」には以下３点のツールの使用権も含まれています。

①Transformation Advisor

- コード分析

②Application Modernization Accelerator

- TAに加え移行ターゲットとして、MoRE(Java, Liberty並行移動対応)、EASeJ(PaaS)追加。分析対象としてMQやDB2が追加。
- 移行のプランニングと優先順位付けを迅速化
- Javaランタイム移行プロセスの分析を最適化するための時間と労力を削減
- 移行先のJavaランタイム構成の構築とカスタマイズに必要な労力を最小化

③Mono2Micro

- モノリシックコードを、マイクロサービスにAIを使い変換

– wCA(IBM watsonx Code Assistant)を使いコードの書き換えを支援する（別売）

生成AIを使い、Java8のコードをJava17へ書き換えを支援する。
https://video.ibm.com/recorded/134415814

TAとAMAの比較

機能	Transformation Advisor	Application Modernization Accelerator
ダウンロード可能なアプリ および構成の分析ツールの対象	WebSphere、 WebLogic、 Tomcat、 JBoss EAP、 およびアプリ単体	TAに加え データベースおよびMQ
ノードグラフによる可視化	n/a	アプリケーション、MQ、DB、 コネクターの 依存関係を可視化
マイグレーション・ターゲット	• WebSphere Base • WebSphere Liberty Core • Open Liberty	TAに加え • Enterprise Application Service for Java (EASeJ) • Modernized Runtime Extension for Java (MoRE)
分析結果に含まれるもの	Reports、 Issues、 Dev Cost、 Common Code、 Guidance、 Recipe Identification	TAと同じ
構成の マイグレーションプランの生成	Liberty、 Container、 Operator(OCP)、 Kustomize	TAに加え • EASeJ、 Db2 SaaS、 MQ SaaS、 IBM Cloud
wCAとの連携	ソースコード変更をサポートする プランをwCAにアップロード可能	TAと同じ

商談前ヒアリングリスト／トークスクリプト

商談前ヒアリングリスト／トークスクリプト

■ 自然な会話で課題を引き出すフロー。アイスブレーク、世間話の挿入時に活用ください。

– 業務のムダ

- 「日常業務で、面倒な手作業や時間がかかる処理はありますか？」
- (質問背景)既に自動化できる仕組みがあっても、「システム間が連携していない」「社内ルールで手作業が残っている」など理由があるはずです。現場の“つぎはぎ運用”がアプリ開発の出発点になることが多くあります。

– 顧客チャネル

- 「顧客との接点（問い合わせ・注文など）は今どのように管理されていますか？」
- (質問背景)電話・FAX・メールなど手段が多岐にわたる場合、集計・確認の手間がかかります。Excelや紙の台帳が残っている場合もアプリ導入の候補です。

– 意思決定速度

- 「経営判断に時間がかかりすぎて困った経験は？」
- (質問背景)情報が集まらない、資料が手元にない、最新情報が共有されていない――これらはシステムの限界が原因のことが多いです。
- データ集約や可視化のニーズを探れます。

– 他社との差別化

- 「競合他社のIT活用で良いと感じた例はありますか？」
- (質問背景)競合との差を感じる瞬間には、本音が出やすいポイントです。ITの活用差、提供スピード、顧客満足度など、アプリ化によって巻き返せる可能性が見えてきます。

– 将来構想

- 「今後追加したい機能や改善したい業務は何ですか？」
- (質問背景)将来的な構想や理想像を知ることで、段階的な導入やPoCから入る提案が可能になります。また、潜在ニーズの把握にもつながります。

提案資料：問題提起型

ミニ知識

■ こんなことにお悩みではありませんか？

- 📄 **労働集約型業務**：手作業・紙中心でムダが多い
- 🕑 **顧客離脱リスク**：対応が遅れ、満足度が下がる
- ⏳ **意思決定の遅れ**：情報が集まらず判断できない
- 🏁 **競争後退**：他社の改善スピードに追いつけない

パッケージやSaaS利用を経て、最終的にアプリ開発に至るケースが多々あります。

なぜ今、“自社開発”が再び注目されているのか？

ミニ知識

■ SaaSでは届かない業務のクセや独自要件

– 汎用ツールでは合わない細かな現場対応が必要

効率化言っても組織をアプリに合わせるのは限界がある。だから開発も捨てられない(ある企業)

■ スピードと柔軟性を両立したい

– 開発によって改善スピードと自由度を確保

■ 既存システムとの連携・再利用

– 一から作るのではなく、今ある資産を活かしたい

■ 将来の成長・拡張を見据えた構造を

– 柔軟な改修・拡張に強い構成で投資対効果を最大化

今ある技術資産をどう活かすか？

ミニ知識

- 自社開発といっても「すべてゼロから」ではありません。SaaSやパッケージと組み合わせましょう。
- 現場には既存のアプリ、業務ロジック、帳票、DBなどが存在。
- それらを**最大限再利用**し、最小限のコストで進化させることが、現代的な開発のポイントです。

→ では、その実現に適した技術とは？

Java資産を活かす意味とメリット

ミニ知識

■ ✅ 今ある仕組みを壊さずそのまま活かせる

– すでに構築されたシステムや設計資産を壊さずに使い続けられます。大幅な作り直しや再教育は不要です。

■ 🛡️ 長年の実績に基づいた安定運用

– 金融・官公庁・大手企業が長年使ってきた実績あり。業務停止リスクが低く、安心して使い続けられます。

■ 👥 対応できる人材が多く、属人化しにくい

– 社内外に知見者が多く、開発・保守・引き継ぎがしやすい。急に誰かが辞めても属人化しにくいです。

■ 🔄 将来の拡張も柔軟に対応（クラウド・自動化）

– 多くのシステムに対応しているため、後からクラウド対応・自動化・データ連携なども柔軟に対応できます。

■ 💰 大幅な再構築をせず、コストを抑えて進化可能

– 「全部作り直す」よりもはるかに安く、安全に機能改善やモダナイズができます。既存投資を無駄にしません。

なぜ「Javaがアプリ開発に適している」と言えるのか？

ミニ知識

- **高い移植性**：どのOSでも動く（JVMベース）
- **長期運用に強い**：安定したエコシステムと実績
- **大規模開発向け**：堅牢な型システムと構造化されたコード
- **人材が豊富**：長年の実績により技術者層が厚い
- **安全性と信頼性**：メモリ管理やセキュリティ対策が標準搭載

なぜ「Javaがアプリ開発に適している」と言えるのか？

ミニ知識

■ 長期運用に強い理由（安定性と保守性）

– **長年使われてきた実績と継続サポート**
  - → 数十年単位で利用され続けており、長期運用でも枯れた（安定した）技術基盤。 例：銀行・自治体システムにおける稼働実績（15年以上）

– **後方互換性が高く、将来のアップデートにも対応しやすい**
  - 「昔作った資産」がそのまま生きる。
  - Java SE 8→11→17 とAPIの互換性が維持されている。

– **ツール群・ドキュメント・知見が豊富**
  - → 保守引継ぎも容易、トラブル対応も迅速。

実在する業種の匿名事例（例：「地方自治体：10年以上Java稼働中」）を挿入

なぜ「Javaがアプリ開発に適している」と言えるのか？

ミニ知識

大規模開発に強い理由

■ 1. 構造化しやすく、チーム開発に強い

- モジュール化・階層化された設計がしやすい
- 複数チームでも分担しやすく、開発がスムーズ
- Springなど業務向けフレームワークも最適化済み

■ 2. 品質を保ちやすい言語仕様

- 静的型付けにより、**バグを開発段階で発見**しやすい
- 型情報がドキュメント代わりになり、**引き継ぎも容易**
- 大規模コードでも**品質・可読性・保守性**を維持しやすい

■ 3. 高い信頼性と安定性

- 仕様変更時も、コンパイルで問題を早期に検出
- 本番でのトラブルを**未然に防ぐ**設計が可能
- パフォーマンス最適化（JITなど）で**実行時も高速**

■ 4. 優れた開発ツールとエコシステム

- IDE支援（Eclipse, IntelliJなどは生成AIとも接続が容易）で**自動補完・安全なリファクタリング**が可能
- 世界中に開発者が多く、**保守・再委託・採用もしやすい**
- 教育体制や教材も豊富、**属人化を防ぐ体制構築が容易**

つまり、？、
(次スライド参照)

なぜ「Javaがアプリ開発に適している」と言えるのか？

ミニ知識

■ 大規模開発に強い理由(型チェック厳格化だけでこれだけよくなる)

- **早期エラー検出**
  - 型チェックで問題を未然に防ぎ、品質を安定化
- **保守性・可読性**
  - 設計意図がコードに明示され、変更に強い
- **開発ツールとの相性**
  - IDEでの自動補完・解析で開発が効率的に
- **実行性能最適化**
  - 実行時の速度と安定性が向上し、応答が速い
- **長寿命設計と品質維持**
  - 継続利用できる構造で技術負債を抑制

Javaに関する人材豊富さの根拠と具体詳細

ミニ知識

■ **人材が豊富**：長年の実績により技術者層が厚い

– **1. 世界のJava開発者数**

- Java開発者は世界で約7～9百万（700万～900万人）と推定されており、主要な言語の中でもトップクラスの普及率を誇ります
- → 世界のプロ開発者人口が約1,960万人のうち、Java開発者が多くを占めている計算です ()。
- inapps「**Developers by Programming Language (2023)**」ではJS, Pythonについで3位。
  - **JavaScript**: 12-13 million (leading language for web development)
  - **Python**: 8-10 million (growing rapidly due to data science/ML applications)
  - **Java**: 7-9 million (strong in enterprise development)
- 「Grid Dynamics」ではJSについで2位。
  - **JavaScript**: 20.0 million (Web, Apps for 3pt ecosystem)
  - **Java**: 17.1 million (cloud, IoT devices)

– **2. 教育・資格制度と技術スキルの担保**

- Oracle社が提供する**Java認定資格（OCA, OCP, Java SE 11/17 Developerなど）**は業界標準となっており、多くの研修機関で採用されています
- → 認定制度が整うことで、新人教育やスキル移転が体系的に可能になります。

– **3. コミュニティとしての支援体制**

- 世界最大級のJavaユーザーグループ「SouJava」（ブラジル）では、**4万人規模**のメンバーが活動しており、地域独自のネットワークと支援ベースを保有しています　。

inapps: https://www.inapps.net/how-many-software-developers-are-in-the-world/?utm_source=chatgpt.com
Grid Dynamics :
https://www.griddynamics.com/blog/number-software-developers-world?utm_source=chatgpt.com

なぜ「安全性と信頼性が高い」と言えるのか？

ミニ知識

■ 強力な型チェックと例外管理機構

– コンパイル時の型チェックや標準的な例外処理モデルにより、**実行時の問題が少なく安定運用**が可能です。

■ バイトコード検証やサンドボックスによる堅牢性

– JVMが安全性を担保する仕組みを内蔵しており、マルウェアや不正コードの影響を抑制します。

■ 大量システムでの実績と脆弱性対応体制

– 金融や行政システム等での導入実績多数。セキュリティパッチや脆弱性対応の体制も確立されており、本番環境での安心感があります。

- **定期的なセキュリティパッチ（Critical Patch Update）の提供**
  - Javaを含む主要製品に対して、Oracleが年4回の定期的なセキュリティ更新（Critical Patch Update＝CPU）を提供しています。
  - → 1月、4月、7月、10月の第3火曜日に必ずリリースされます。
  - 実例として、2025年1月には**318件のセキュリティ更新（うち30件が重大な脆弱性）**が提供されました。()
- **広範な脆弱性への迅速な対応と実績**
  - 2024年4月のパッチでは、239件のCVE（脆弱性識別番号）に対応する441件の修正が適用され、そのうち8.6%が重大脆弱性でした。
  - https://www.tenable.com/blog/oracle-april-2024-critical-patch-update-addresses-239-cves
  - また、前回（2025年1月）には186件のCVEに対して318パッチが提供され、9.4%が重大脆弱性。
  - https://zh-tw.tenable.com/blog/oracle-january-2025-critical-patch-update-addresses-186-cves

ライセンス関連

WebSphereのライセンス（三つのエディション）

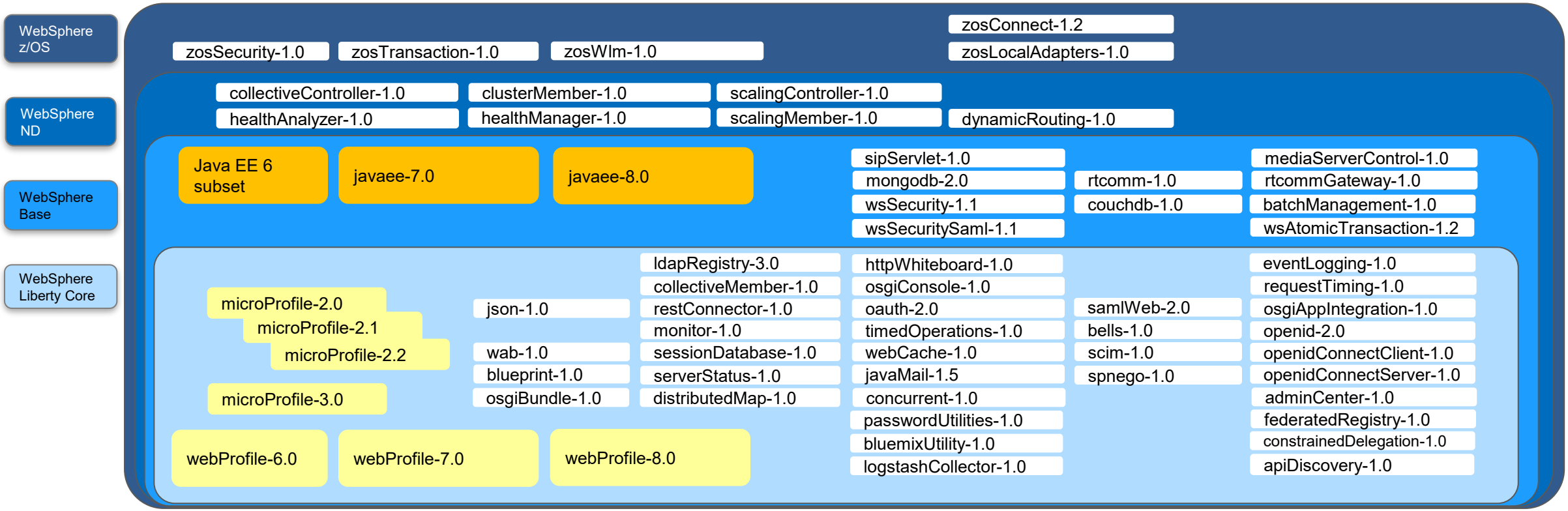
エディション	提供機能	Liberty ランタイム	traditional ランタイム
WAS ND	➢ Java EE Full Platform対応 • 複数サーバーの集中管理機能 • クラスタリング、高可用性、拡張性、運用最適化	WAS Liberty Java EE 7/8 完全対応 MicroProfile対応	WAS traditional Java EE 7 完全対応
WAS Base	➢ Java EE Full Platform対応 • サーバー単位での管理 • シンプルな割振り構成	WAS Liberty Java EE 7/8 完全対応 MicroProfile対応	WAS traditional Java EE 7 完全対応
Liberty Core	➢ Java EE Web Profile対応 (利用できるAPIが限定される) • サーバー単位での管理 • シンプルな割振り構成	WAS Liberty (Core) Web Profile 7/8 対応 MicroProfile対応	N/A

tWASからLibertyへ乗り換え可能ですが、どちらかだけ使えます。

WebSphere Libertyで提供されている機能

これらのFeatureが従来のアプリケーションサーバーで提供されていた機能

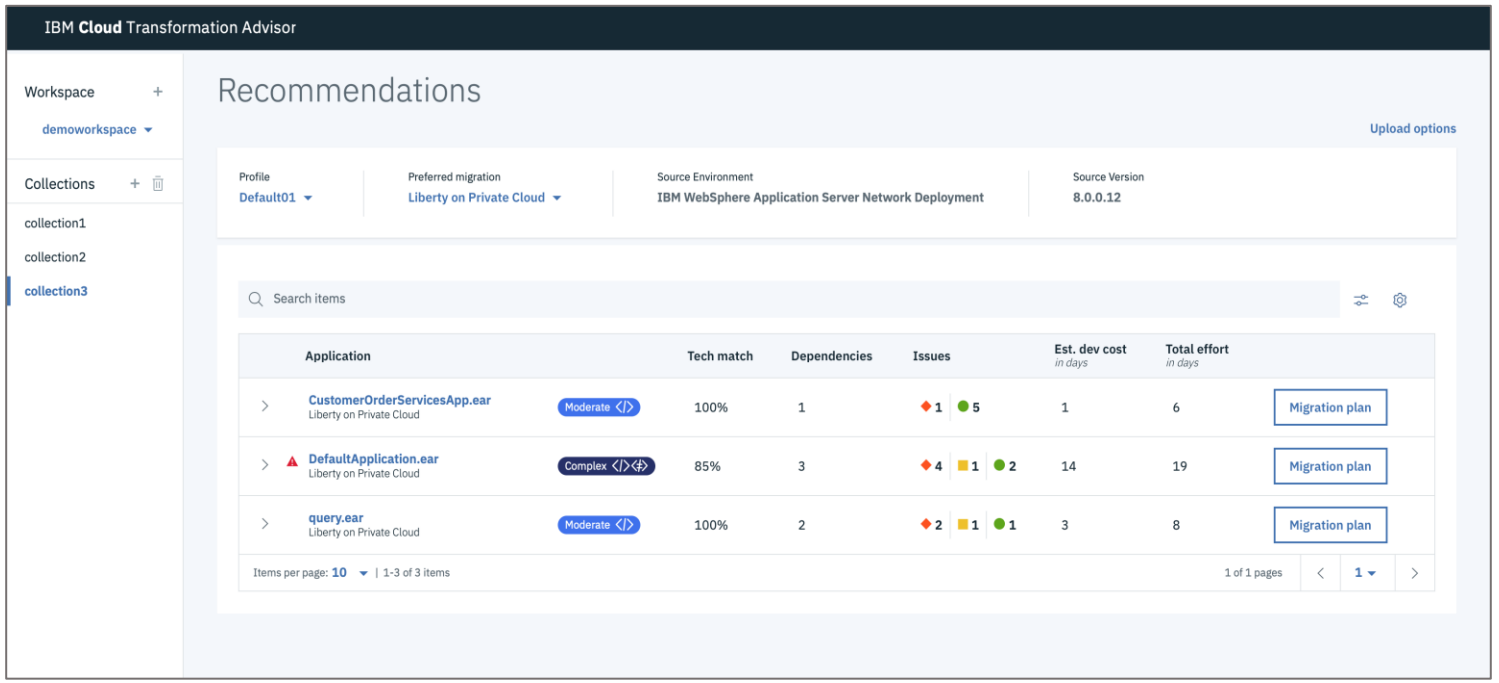
- 不要ならばOffにできる
- WebSphere Baseエディションでは最初から提供されていない



TA：WAS traditionalや他社製品からLibertyへの移行に

■ Transformation Advisor

- 既存アプリケーションサーバー環境/アプリをスキャン
  - WebSphere、 WebLogic、 JBoss、 Tomcat をサポート
  - アプリケーションのみであれば 国産ベンダー環境でも
- マイグレーションの考慮点整理、レポート生成
- コンテナ化に必要な一連のファイルの自動生成
  - Dockerfile、 Liberty構成ファイル（server.xml)など
- コードをレシピに従って自動修正するOSS、OpenRewriteと連携



Simple

そのままコンテナ化可能

Moderate

一部コード修正など対応が必要

Complex

一部アプリケーションの書直しが必要

https://ibm.biz/TransAdv

Migration Toolkit：移行の調査を簡便に

■ Migration Toolkit for Application Binaries

アプリケーション評価レポート

ご使用のアプリケーションは、以下の IBM プラットフォームが提供するテクノロジーを使用します。

WebSphere Application Server V9.0

	IBM Bluemix 上の Liberty for Java	Liberty Core	Liberty	WebSphere traditional	Network Deployment Liberty	Network Deployment traditional	Liberty for z/OS	WebSphere traditional for z/OS
WEB サービス・テクノロジー								
Java API for XML-based RPC (JAX-RPC)				✓		✓		✓
SOAP with Attachments API for Java (SAAJ) 1	✓	✓	✓	✓	✓	✓	✓	✓
WEB アプリケーション・テクノロジー								
Java Servlet	✓	✓	✓	✓	✓	✓	✓	✓
JavaServer Pages/Expression Language (JSP/EL)	✓	✓	✓	✓	✓	✓	✓	✓
エンタープライズ・アプリケーション・テクノロジー								
Enterprise JavaBeans (EJB) 2.x および 1.x	✓		✓	✓	✓	✓	✓	✓

分析したこのアプリでは古いAPI（JAX-RPC）が使用されているので
Libertyは使用できず、WAS traditionalでしか実行できない

- TAのアプリケーション分析でも利用されているツール
- アプリのEAR/WARを直接分析し、使用されているAPIや機能を調査
- アプリケーションがWAS Liberty WAS traditionalで実行することができるか分析
- Liberty Coreエディションが利用可能かも調査可能
- 修正や注意が必要な点を指摘
- ソースやが不要で、サーバーでのツール実行も必要なく簡易に調査が実施できる

https://www.ibm.com/support/pages/migration-toolkit-application-binaries

Migration Toolkit：移行の調査を簡便に

Migration Toolkit for Application Binaries 使用方法

- コマンドラインから以下のように実行
  java -jar binaryAppScanner.jar <WAR/EAR>
  --sourceJava=[oracle5|oracle6|oracle7|oracle8|ibm7|ibm8]
  --sourceAppServer=[weblogic|jboss|tomcat|was855|was80|was70]
  --targetJava=[ibm8|java11|java17|java21]
  --targetAppServer=liberty
- アプリケーションで使用されているAPI
  修正が必要な点、アプリの構成
  などを分析し結果を出力します
- デフォルトでは結果は
  HTMLファイルとして出力

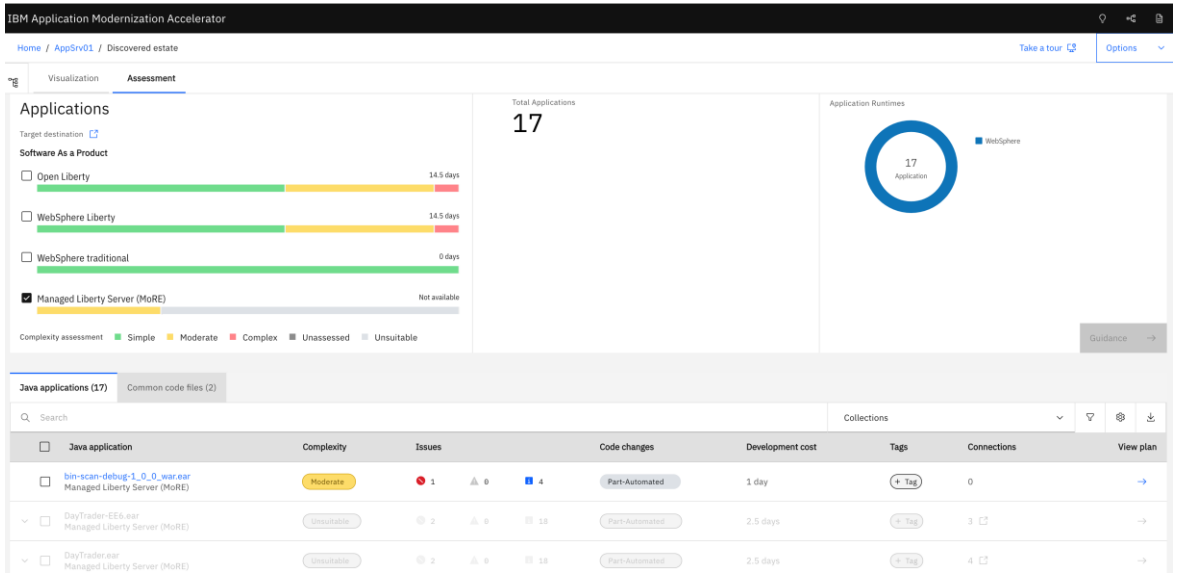
- 既存のWAS v8.5/v9.0にたいして
  最新Fixpackを適用すると、
  管理コンソールから同分析を実行可能

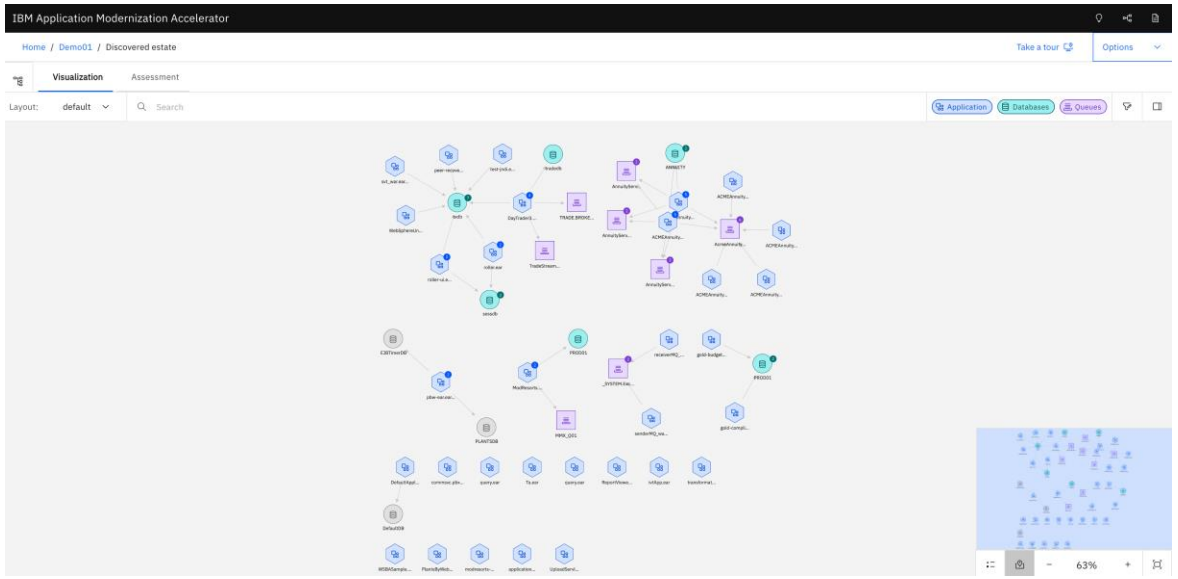
AMA : Application Modernization Accelerator

- AMAは、コンテナ上で実行されるTAを強化・拡充した新しい移行ツール

- AMAはJavaランタイムのアプリケーションモダナイゼーションを強力にサポート
  - エンタープライズJavaアプリケーションとDb2、MQなどの依存関係を検出
  - 分析レポートを作成（コードの変更、複雑さ、移行コストを特定）
  - モダナイゼーションのための移行計画を生成
  - 新しいJavaランタイムのアーティファクトを自動生成

- 価値
  - 移行計画と優先順位付けを迅速化
  - Javaランタイム移行プロセスの分析を最適化するための時間と労力を削減
  - 移行先に合わせたJavaランタイム構成の構築とカスタマイズに必要な労力を最小限に





TAとAMAの比較

機能	Transformation Advisor	Application Modernization Accelerator
ダウンロード可能なアプリおよび構成の分析ツールの対象	WebSphere、 WebLogic、 Tomcat、JBoss EAP、 およびアプリ単体	TAに加えデータベースおよびMQ
ノードグラフによる可視化	n/a	アプリケーション、MQ、DB、 コネクターの依存関係を可視化
マイグレーション・ターゲット	• WebSphere Base • WebSphere Liberty Core • Open Liberty	TAに加え • Enterprise Application Service for Java (EASeJ) • Modernized Runtime Extension for Java (MoRE)
分析結果に含まれるもの	Reports、 Issues、 Dev Cost、Common Code、 Guidance、Recipe Identification	TAと同じ
構成のマイグレーションプランの生成	Liberty、 Container、 Operator、Kustomize	TAに加え • EASeJ、 Db2 SaaS、 MQ SaaS、 IBM Cloud
wCAとの連携	ソースコード変更をサポートするプランをwCAにアップロード可能	TAと同じ

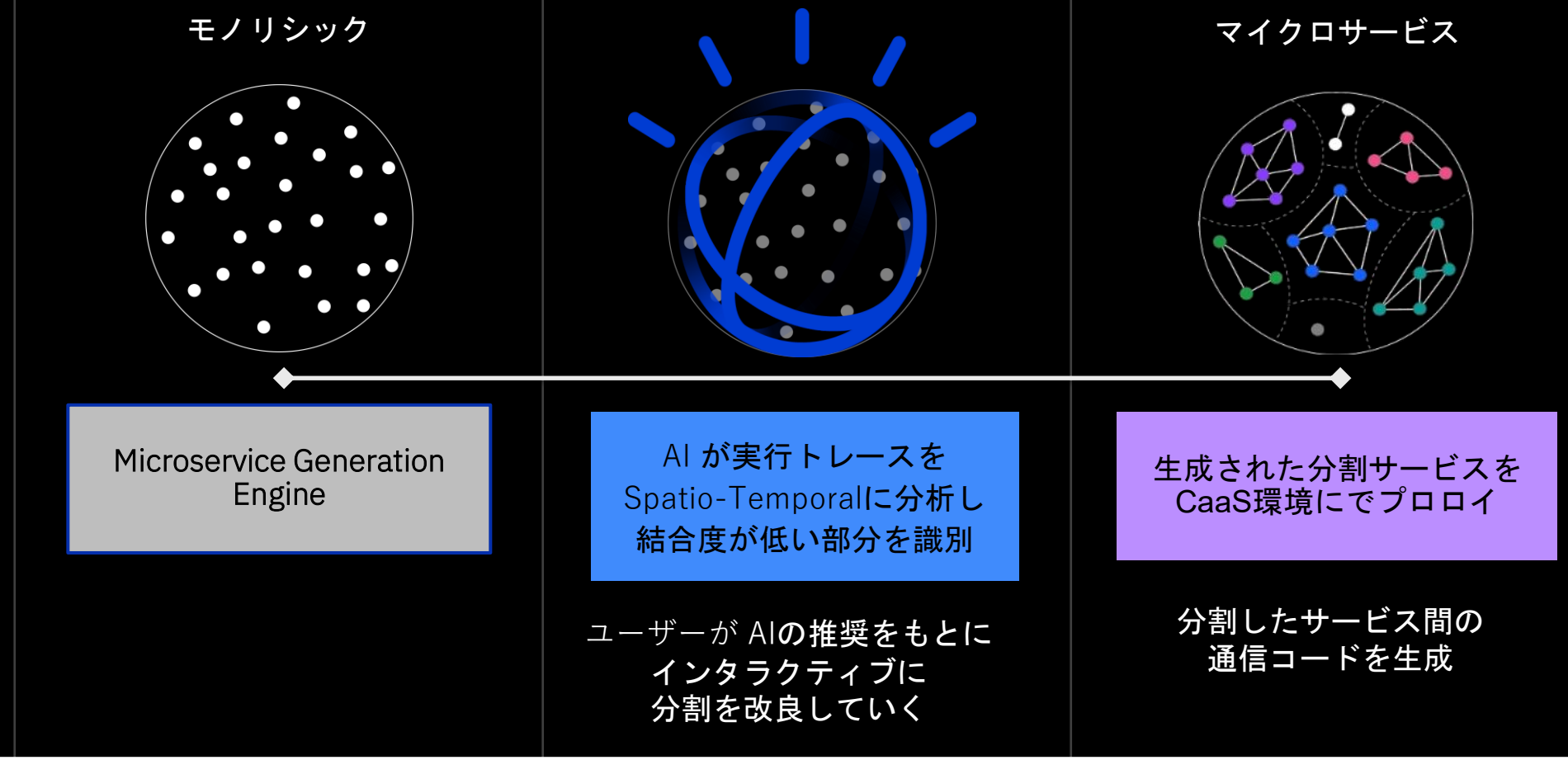
Mono2Micro

- AI技術を活用して、モノリシックなアプリをマイクロサービスに変換

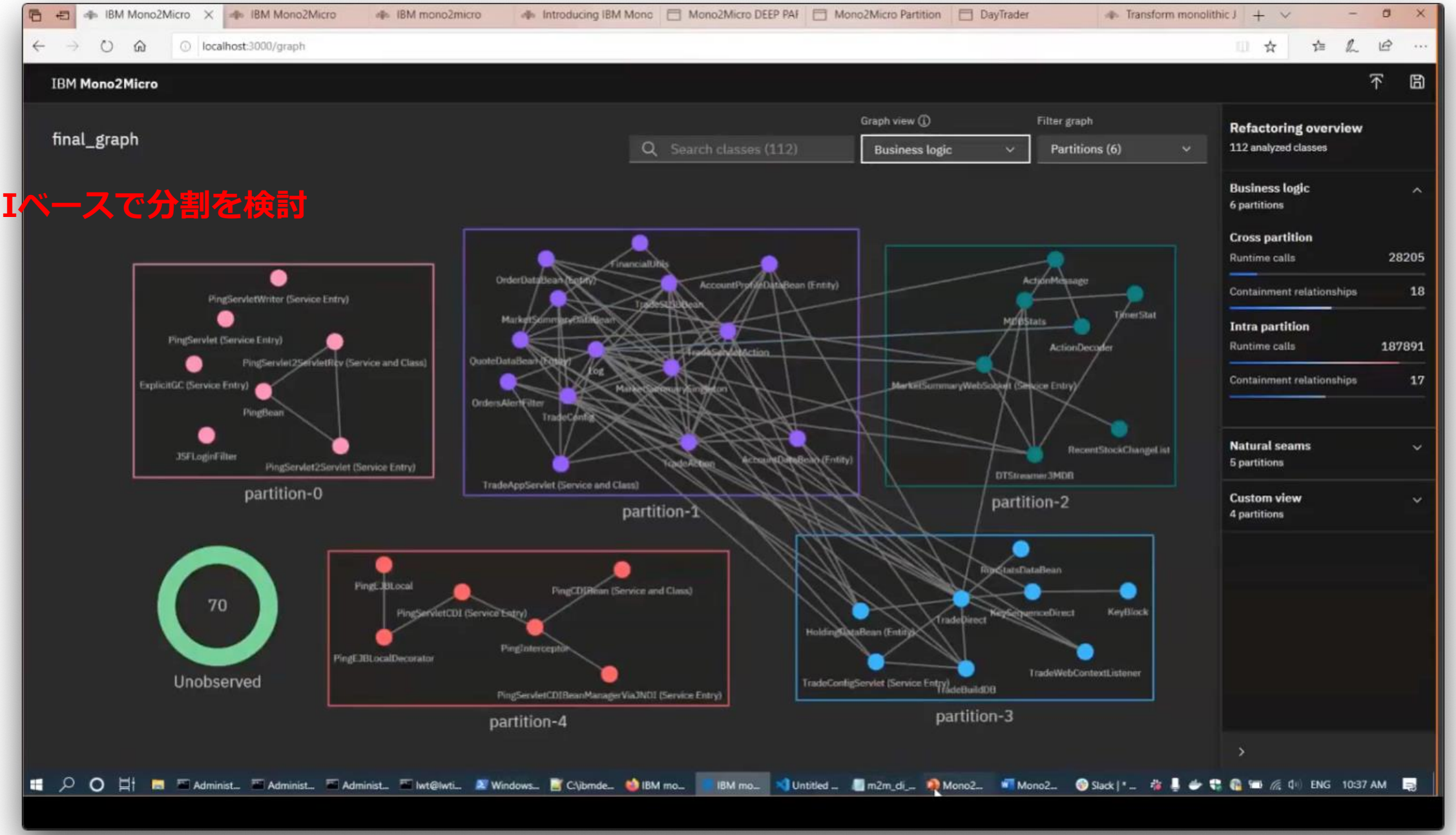
http://ibm.biz/Mono2Micro

無償で90日間試用可能

Mono2Microは、
AIによってアプリの
実行トレースを
時間的・文法的に
分析し、分割の
推奨を作成し、
リファクタリングに
必要なコードの
大部分を生成します

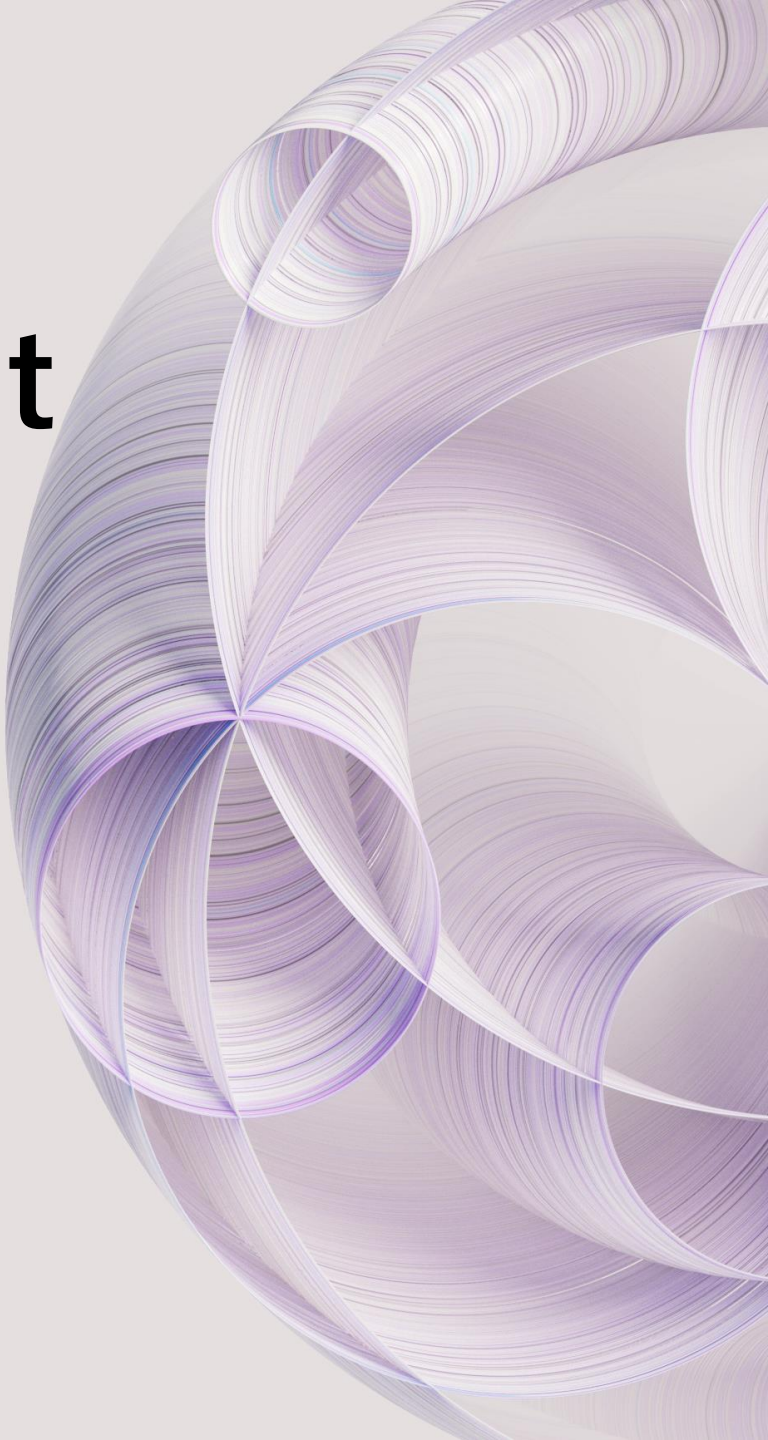


GUIベースで分割を検討



IBM watsonx Code Assistant のご紹介

汎用コード開発 + Javaモダナイゼーション



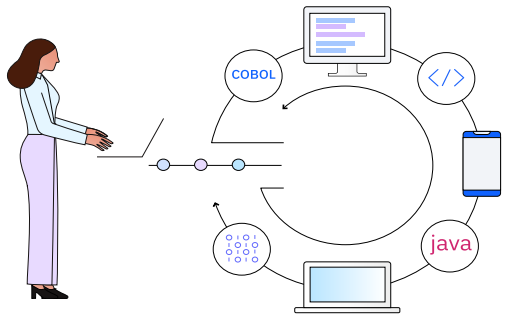
テクノロジー事業本部
データ・AI・オートメーション事業部

watsonx Code Assistantポートフォリオ

スキル・ギャップに対応し、開発者の生産性を向上させるための
企業対応のコード・ソリューション用AI

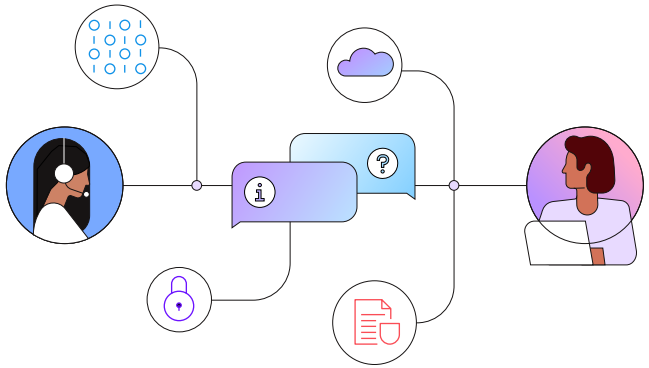
1. 最先端のIBM Graniteモデルを搭載

- プログラム言語と構文を包括的に理解し、高度に分類されたデータで学習されました
- コード生成、コード説明、エラー検出において、同等のパラメーターを持つ他のコードモデルを上回るパフォーマンス



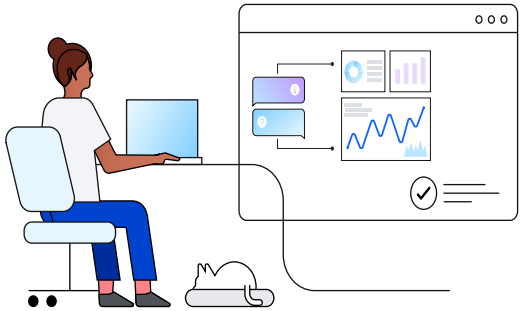
2. コーディングだけでないアシスタントツール

- 生成AIと、業界をリードする最適な自動化ソリューションを組み合わせてアプリケーションライフサイクルを加速します
- 最適なエンドツーエンドの開発者体験のための深い統合と相互運用性により、スピードと俊敏性を実現します



3. 信頼性と透明性の高い設計

- データの出所、リスク、コンプライアンスに焦点を当てた厳格なデータガバナンスプロセス
- IBMは、IBMが開発した基盤モデルに対する第三者の知的財産権請求について、お客様を保護します



コミュニティ加入促進

今すぐ試してみたい方は：Open Liberty

Open Source Softwareとして開発され無償で利用いただけます。
IBMサポートをおつけすることも可能なので、安心して本番環境でお使いいただけます。

- 軽量・高速でクラウド・ネイティブ、コンテナに適したJavaアプリケーションランタイム
- オープンソースとして開発 / GitHub上でソースを公開・問題管理
  https://openliberty.io/ https://github.com/openliberty/
- 無償で利用可能 / 商業利用しやすいEPL（Eclipse Public License）で公開
- Java EE/Jakarta EE/MicroProfileなどのAPIは、
  WebSphere Liberty製品版（Baseエディション）と同等の機能を提供
- 新機能追加を含むバージョンは製品版と同時にリリースされる
- IBM Cloud Pak for ApplicationsライセンスでIBMサポートの提供が可能

Open Liberty

https://openliberty.io/start/

手元ですぐに試せるハンズオン

■ 「Visual Studio CodeとLiberty Toolsではじめる Open Liberty開発 最初の一歩」
– https://qiita.com/TTakakiyo/items/5f09be651bec34885c6e

■ 「1時間でなんとなくわかった気になるOpen Libertyでアプリケーション開発
& コンテナビルド & OpenShiftへデプロイ」
– https://qiita.com/babatch324/items/38d0bfdd18ebdb5bd906

コミュニティで最新情報をチェック！

- 多数の日本語技術文書を掲載
- 各種イベントやセミナーのご案内
- IBMidでLoginすれば質問などを登録することも可能

–https://ibm.biz/JapanWebSphereUG

Japan WebSphere
User Group (JWUG)

IBM TechXchange、 Japan WebSphere User Community

移行の悩みをズバリ解決！
「JavaEE→JakartaEE」最新情報セミナー

日時　：2025/8/7（木）17:00-18:00
場所　：オンラインのみ
対象　：お客様、ビジネス・パートナー様、その他
参加費：無料

今年前半には、エンタープライズJavaの標準仕様、Jakarta EE 11の公開が予定されています。Spring Frameworkや各社のアプリケーションサーバーなども最新版からはJakarta EE対応となり、いよいよJava EEからJakarta EEへの移行を考えられているお客様が多いかと思います。Java EEからJakarta EEへの移行は、提供されているAPIの名前空間（パッケージ名）が変更になったため、大規模な書き換えや関連ライブラリのバージョンアップなどが必要になり、負担の大きいプロジェクトになりがちです。
このセッションでは移行の負荷を軽減する各種のツールをご紹介するほか、どうしても移行できないアプリケーションの安全な延命策についてもご案内します。

アジェンダ（予定）

- エンタープライズJava仕様（Jakarta EE）の現状
- 移行にあたって必要な名前空間（パッケージ名）の変更について
- 移行ツールのご紹介
- 移行できないアプリケーションの延命策：
  Java EEのAPIのサポートを続けるOpen Liberty / WebSphere Liberty
- Libertyにより可能になるITシステムのモダナイズ

Register today →

URL：https://ibm.biz/BdnmA9
IBMグループ社員はこちら：https://ibm.biz/Bdnwvi