WebSphere Application Server V9
アナウンスメント・セミナー

WAS Liberty

日本アイ・ビー・エム株式会社
クラウド・ソフトウェア事業部
アプリケーション・プラットフォーム
斎藤 和史

アジェンダ

- WAS Libertyの特長
- Create: **クラウド・ネイティブ**
- Connect: **API機能強化**
- Optimize: **プロダクション・レディー**

WAS Liberty とは

- WAS V8.5 (2012年6月) から提供している軽量なアプリケーション・サーバー・ランタイム
  - 軽量・高速・簡単構成
  - (OSSを含む) 自動化ツールとの連携
  - Java EE 7 完全対応
  - …

Libertyは，変化の速い，これからのアプリケーション開発・運用に対応したランタイム

WAS Libertyの特長

①Java EE 7対応
Java EE 7 標準に準拠したアプリを完全サポート
JAX-WS, JAX-RS, JMSもサポート
新機能も継続的に提供

②軽量ランタイム
メモリー使用量が小: 60MB程度
ディスク使用量も100MB程度
起動が速い：5秒程度

③Unzipによる導入とデプロイ
パッケージをした
サーバー ＋ アプリ ＋ 構成情報を
Unzipでデプロイ可能

④簡単な構成と動的変更
最低限必要な構成ファイルはserver.xmlひとつだけ
デフォルトベースで簡単構成
構成変更は再起動なしに反映

WAS Liberty & WDT

⑤統合ツール（WDT)
高機能なEclipse用の連携ツール
を無償で提供
Eclipseから簡単に使用可能

⑥自動化ツールとの連携
多くのOSSツールに
無償でプラグインを提供

⑦様々な環境で稼動
オンプレ，クラウド(IaaS、
PaaS)，Dockerで稼動

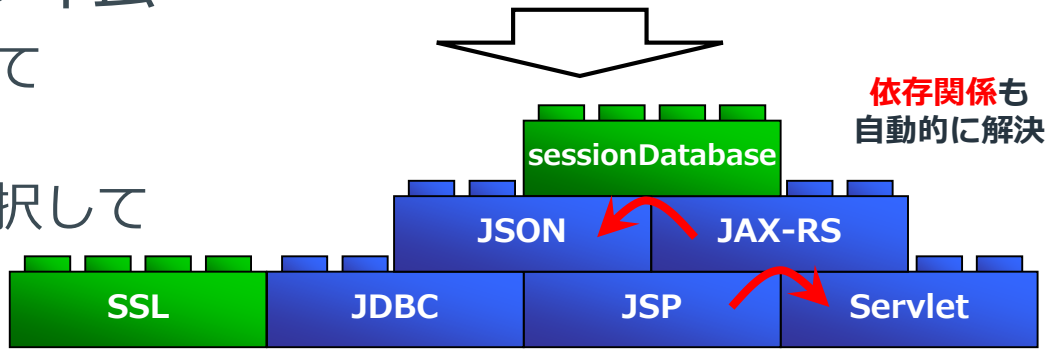
⑧API公開
RESTのアノテーションから
Swaggerを自動生成

軽量・高速なランタイム

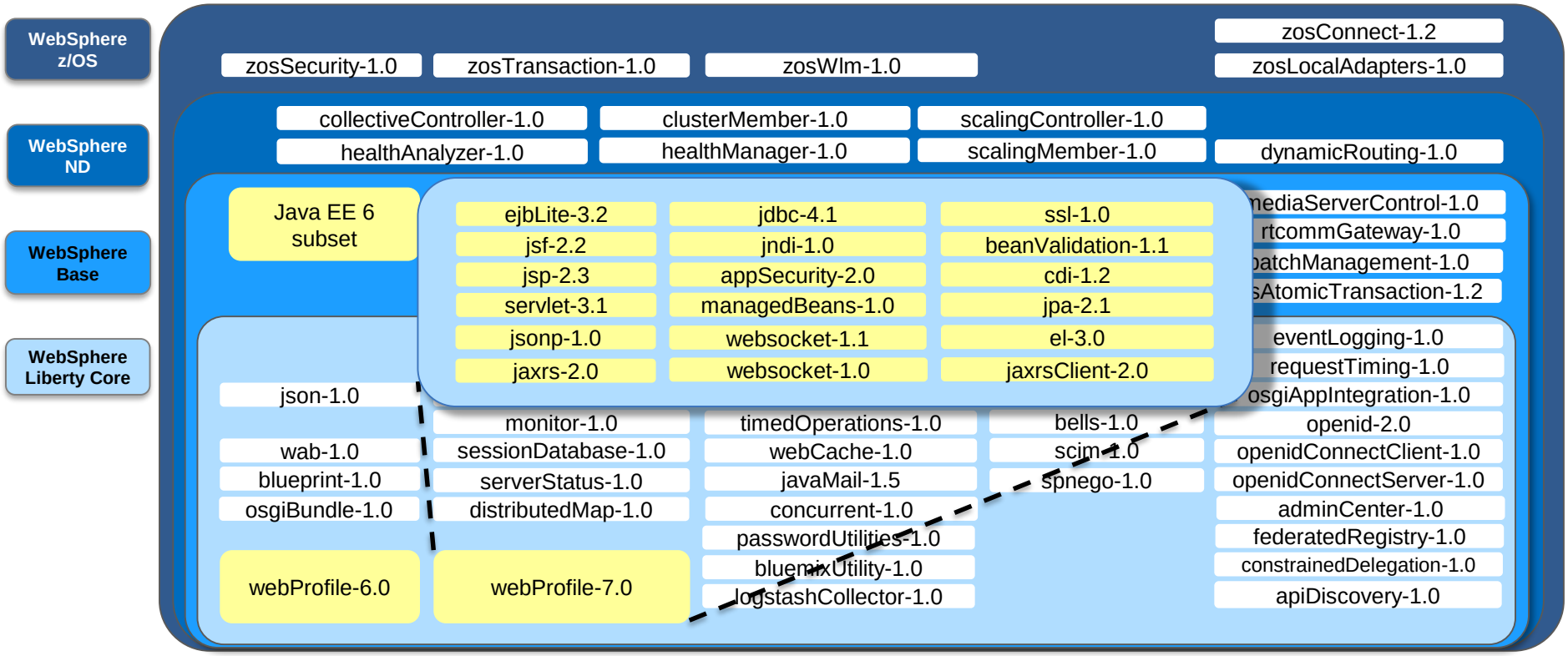
- 軽量
  - 数十メガバイトのメモリ消費／100メガバイト程度のディスク消費
  - コンテナや仮想環境への集約が容易に
- 高速起動・動的変更
  - 数秒以内でサーバーが起動
  - サーバーの構成変更やアプリケーションの変更も即座に反映
- モジュール構造のランタイム
  - 提供機能をFeatureとしてモジュール化
  - 必要なFeatureだけを選択して導入・起動

構成ファイル server.xml

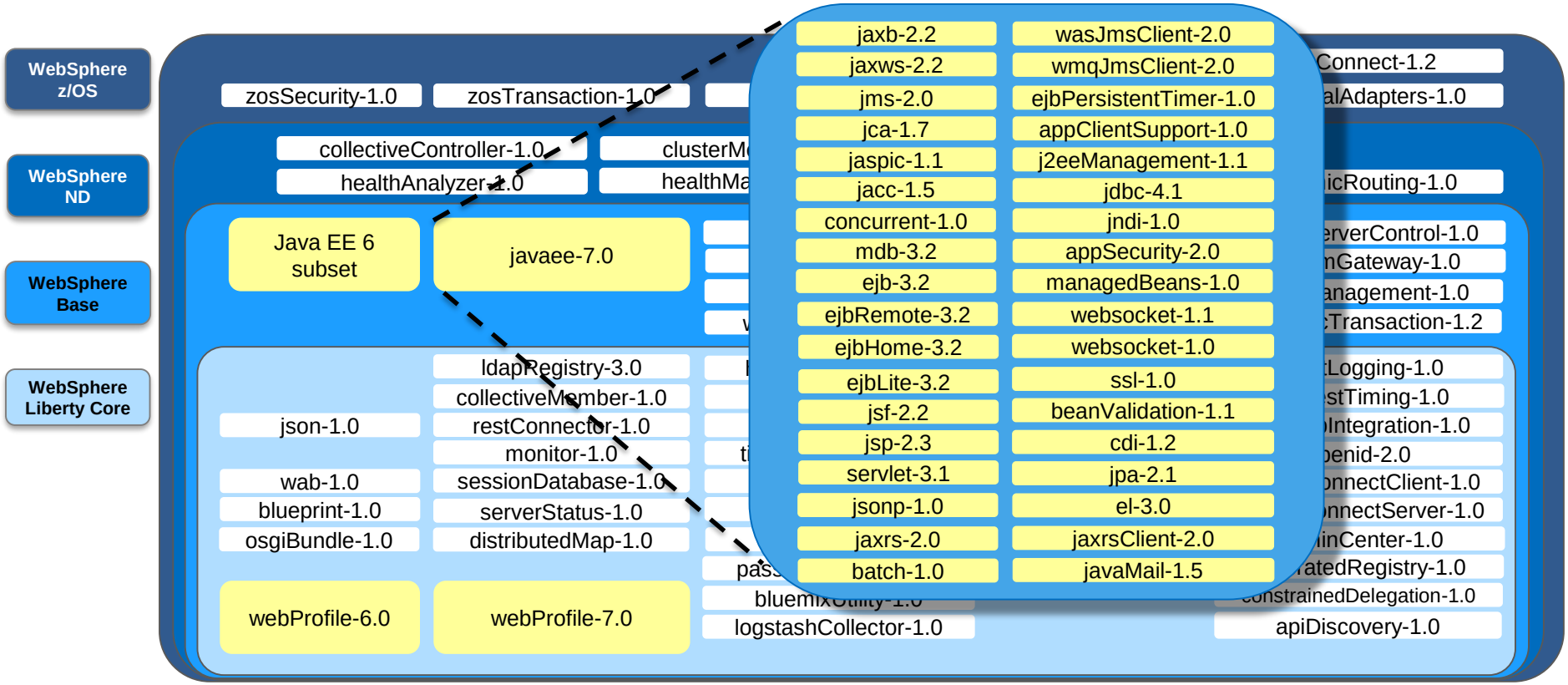
```
<featureManager>
    <feature>jsp-2.3</feature>
    <feature>jdbc-4.1</feature>
    <feature>jaxrs-2.0</feature>
    <feature>sessionDatabase-1.0</feature>
    <feature>ssl-1.0</feature>
</featureManager>
```



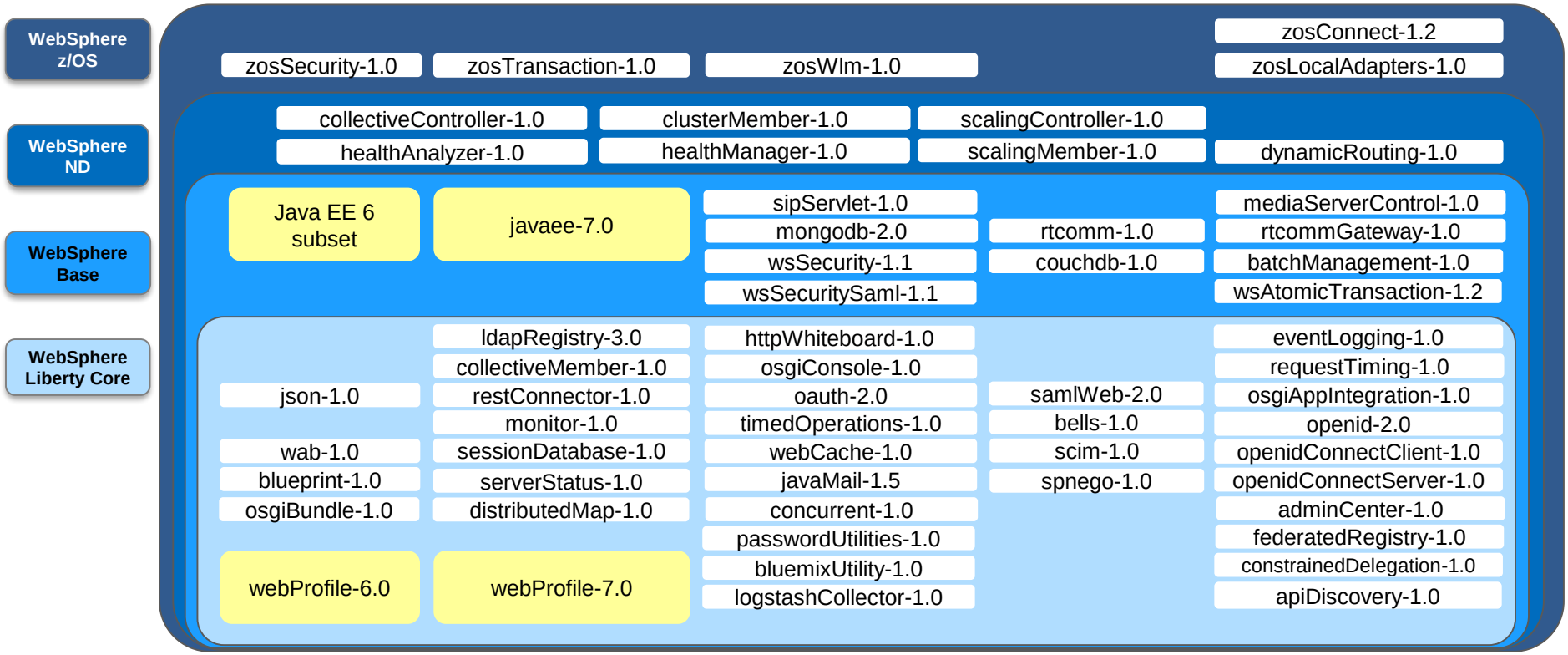
Java EE Web Profile



Java EE Full platform



Liberty独自のフィーチャーも



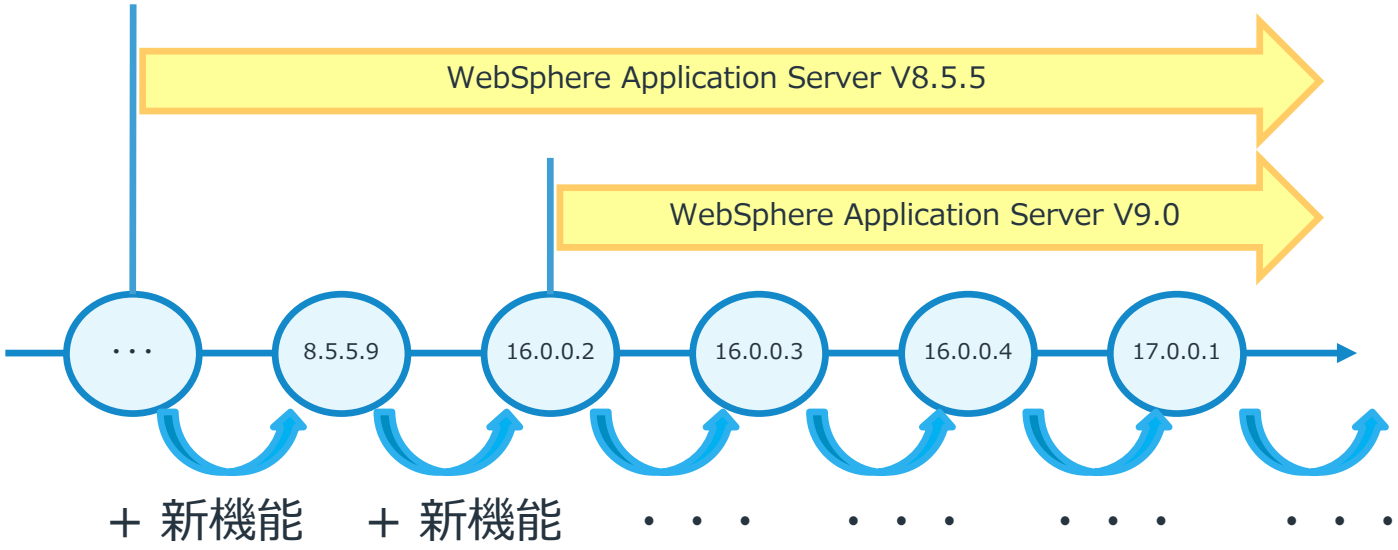
継続的デリバリー・モデル

- 継続的デリバリー・モデルで新機能を提供
  - 四半期ごとに新機能を取り込んだフィックスパックを提供

16 . 0 . 0 . 2

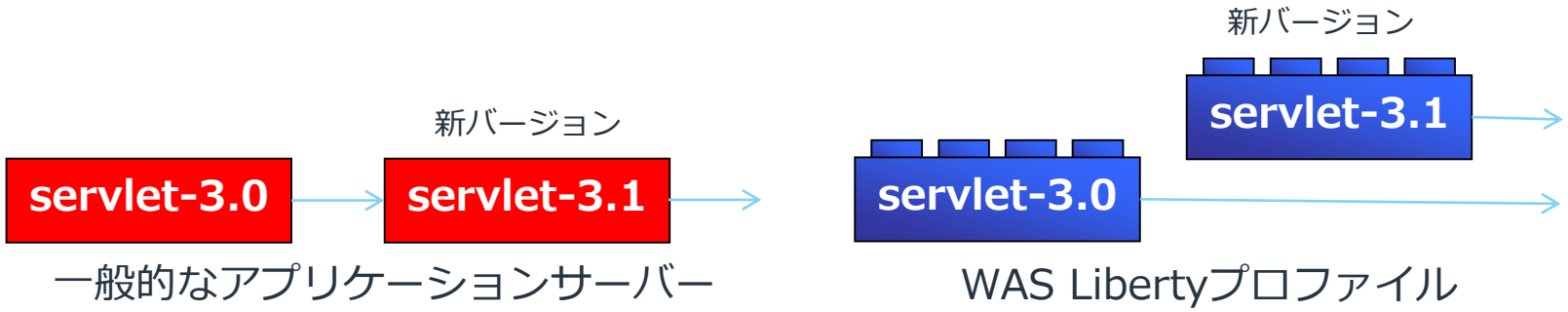
Year　Release　Modification　Fix

2016年に出荷した，2016年内の2つ目のフィックスが適用されたLiberty



ゼロ・マイグレーション

- 新しい仕様のバージョンに対応するフィーチャーが追加されても，従来のバージョンも提供
  - 例） 現在はServlet 3.1フィーチャーが提供されていますが，Servlet 3.0フィーチャーも引き続き利用できます



- アプリケーションで新仕様が必要なければ従来のフィーチャーをそのまま利用可能
  - ただし，対応するJDKのバージョンは変更される可能性があります

Create

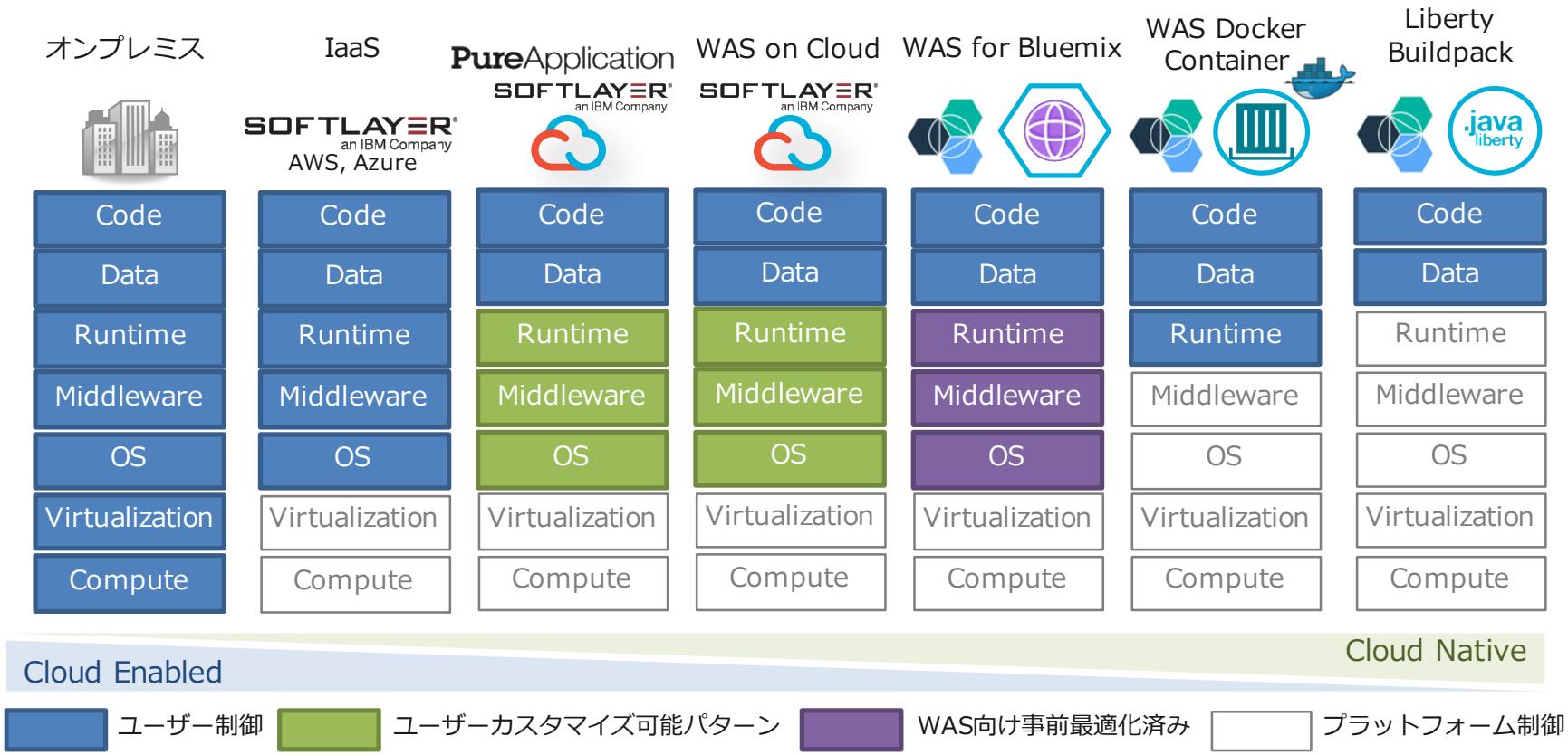
Libertyでクラウド・ネイティブ

Libertyはクラウド・ネイティブを強力支援

- あらゆるクラウド環境で稼働
- Bluemix (PaaS)なら，アプリをデプロイするだけ
- 開発用にDocker，さらに本番でもDocker
- Libertyでアプリ開発を支援するコンテンツ

多様なクラウドとの親和性

WAS Libertyは，オンプレミスだけでなく，IaaS，PaaS，コンテナ，など幅広い環境で稼働します。軽量・高速・簡単であり，可搬性の高いアプリケーション・サーバーです。



Bluemix上の3つのLibertyプロファイル

IBM Bluemix Ready? Try the new Bluemix | New! Try OpenWhisk

ソリューション　カタログ　料金　資料　コミュニティー

検索の絞り込み:

liberty

- スターター
  - ボイラープレート
- 計算
  - ランタイム
  - コンテナー
- サービス
  - Watson
  - モバイル
  - DevOps
  - Web とアプリケーション
  - ネットワーク
  - 統合
  - データおよび分析
  - セキュリティー
  - ストレージ
  - ビジネス・アナリティクス
  - モノのインターネット
  - カスタム API
- プロバイダー
  - IBM
  - サード・パーティー

計算 // Cloud Foundry または Docker イメージから開始します

ランタイム

選択した言語でアプリを実行

Liberty for Java™
IBM

1）Liberty for Java

コンテナー・イメージ

IBM イメージからコンテナーを作成するか、独自のイメージを追加します

ibmliberty
IBM

2）Libertyコンテナ

サービス // 卓越したすべてのアプリのビルディング・ブロック

Web とアプリケーション

新しい Web アプリとモバイル・アプリのデリバリー

WebSphere Application Server
IBM

3）WAS for Bluemix

Bluemix: **Liberty for Java**

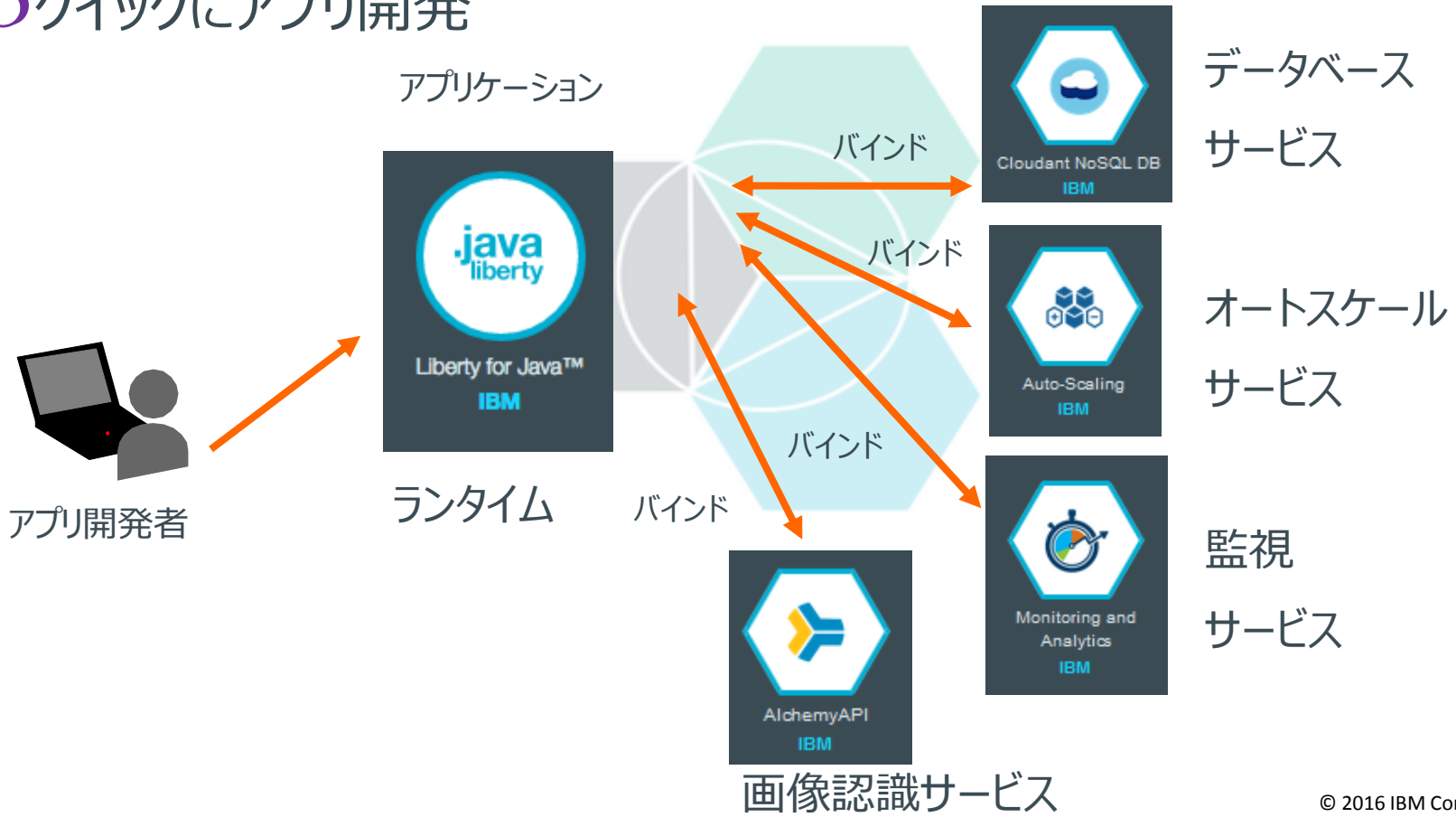
- Bluemix上で提供するJavaアプリケーション実行環境
- 製品版のWAS Libertyプロファイルと同じ実行環境
  - .warをデプロイするだけで，
    一般的な**Java EEアプリケーション**がそのまま稼働

Launching defaultServer (WebSphere Application Server 16.0.0.2/wlp-1.0.13.cl160220160526-2
J9 VM, version pxa6480sr3-20160428_01 (SR3) (en_US)

既に 16.0.0.2

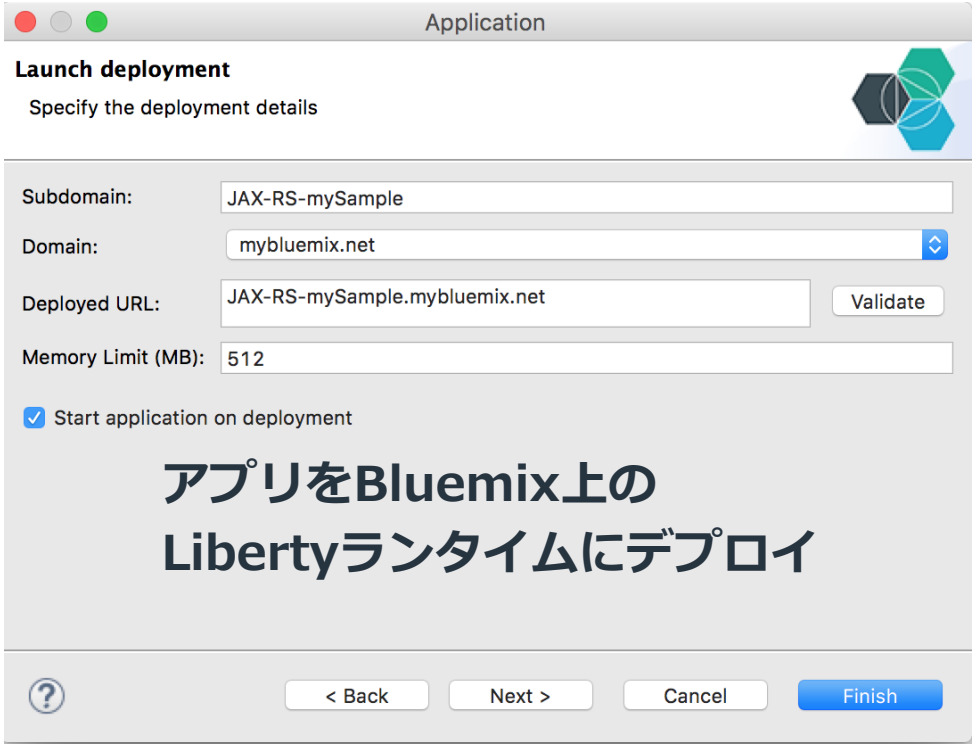
Bluemixサービスをバインド(連携)

- 欲しい機能を選んで組み合せる
  - 100を超える高品質のサービスを活用して，
  - クイックにアプリ開発

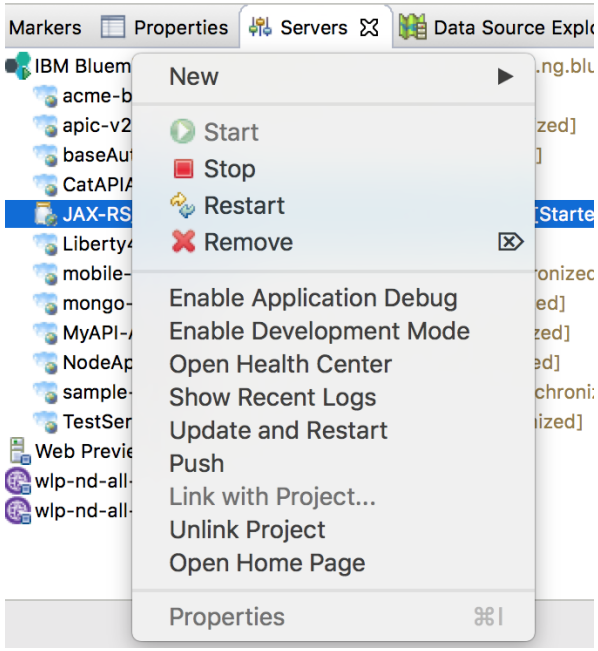


Bluemixプラグイン (Eclipse)

- Eclipseでアプリを開発し，そのままLibertyランタイム(Bluemix上)へのデプロイが可能



アプリをBluemix上のLibertyランタイムにデプロイ



- EclipseからBluemix上の各アプリの制御が可能
  - 始動・停止
  - デバッグモード
  - ログ閲覧，etc.

Bluemix: **Libertyコンテナ**

- Bluemix上で提供するDockerコンテナ環境
  - 複数コンテナ構成，負荷分散
  - スケーリング，自動リカバリ
  - ロギング，モニタリング

- プライベート・リポジトリ
  - IBM提供イメージの利用

任意のDockerイメージを
選択して，コンテナ作成

  - 独自のイメージをpushして利用するのも可

Bluemix: **WAS for Bluemix**

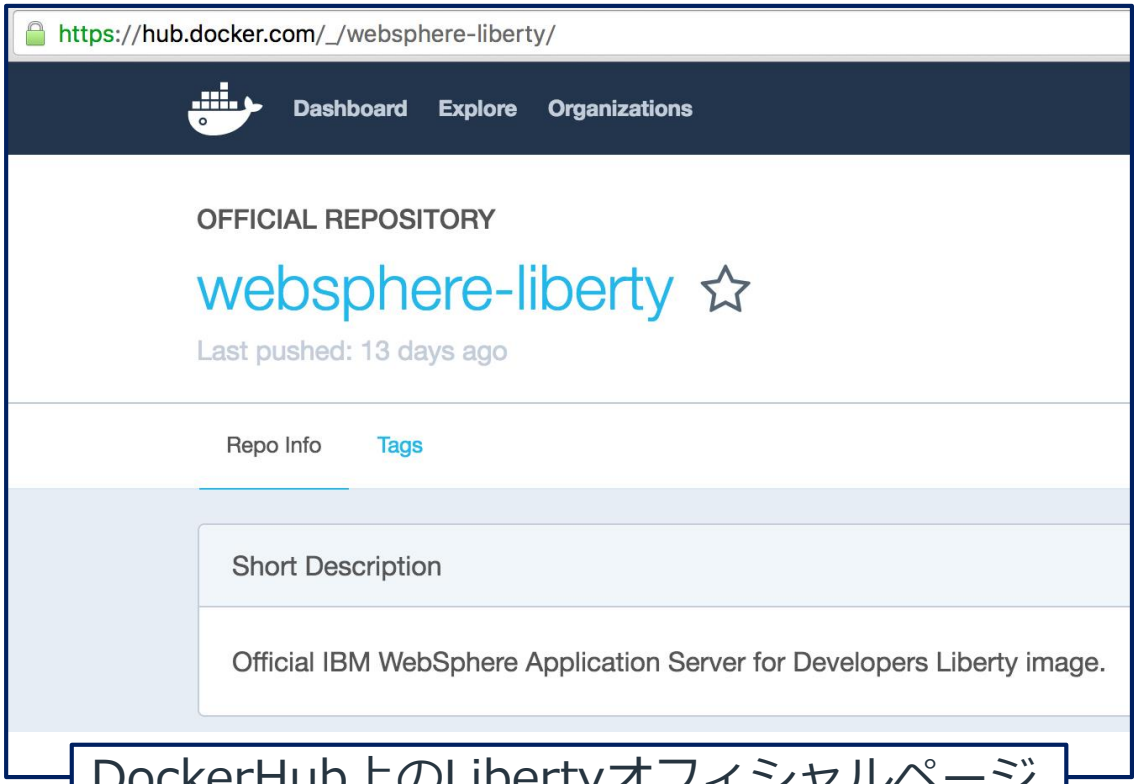
- Bluemix上で提供するVMサービス
- Libertyを事前構成済で提供
  - シングルサーバー構成
  - Collective構成
- VM内のOS設定も可能
- AdminCenterへのアクセス

	vCPU	Mem	Disk
S	1 vCPU	2 GB	12GB
M	2 vCPU	4 GB	25 GB
L	4 vCPU	8 GB	50 GB
XL	8 vCPU	16 GB	100 GB
XXL	16 vCPU	32 GB	200 GB

5種類から選べるリソースサイズ

Liberty Docker

- DockerHubに公開
  - 開発版Libertyイメージ(無償利用可)
  - Dockerfile群
    - webProfile6/7
    - javaee7など



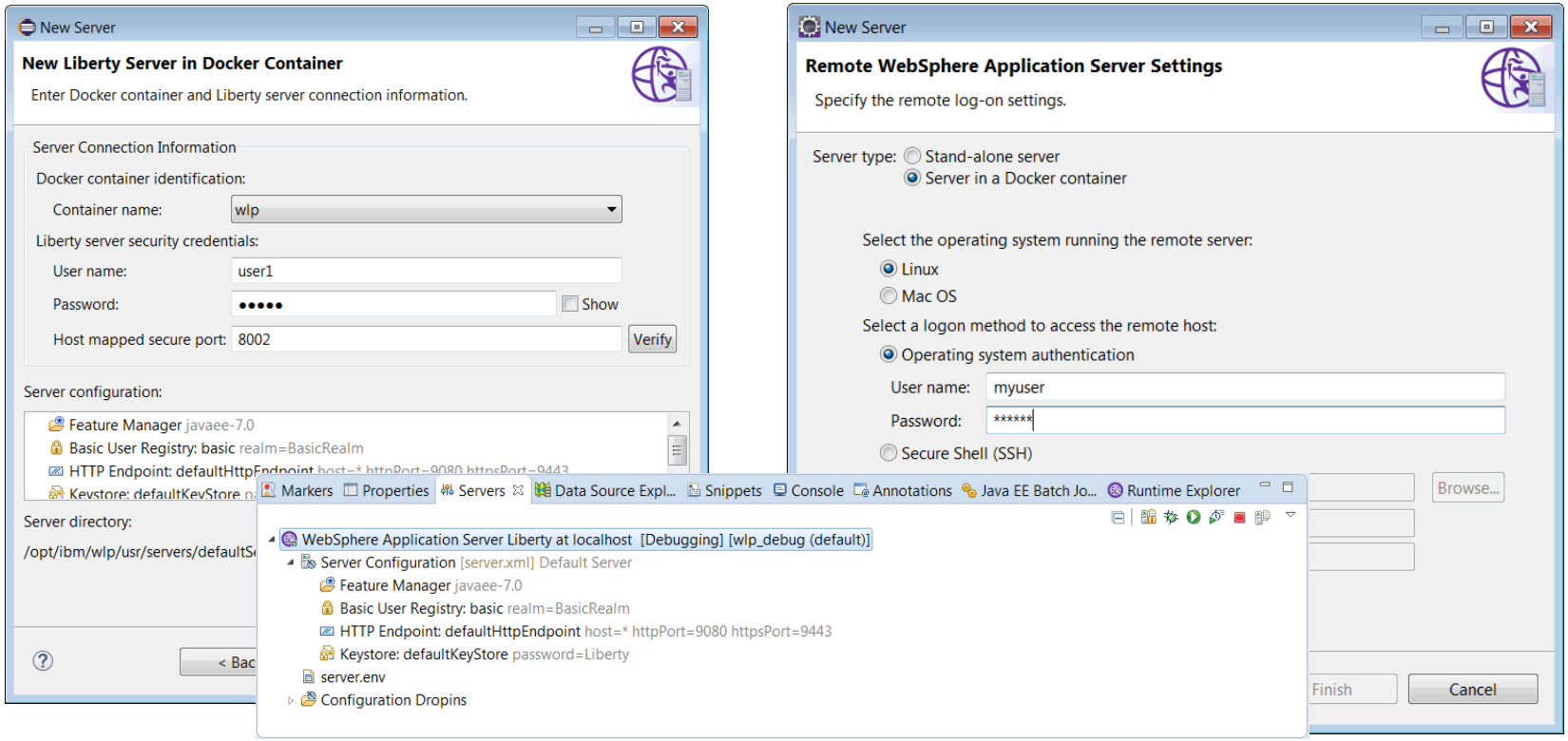
DockerHub上のLibertyオフィシャルページ

- 本番利用
  - 製品版バイナリーから本番イメージ作成
  - 開発版にライセンス化ファイル適用

- テスト済クラウド環境
  - IBM Containers(Bluemix)
  - Docker Datacenter
  - OpenShift V3

WDTもDocker対応

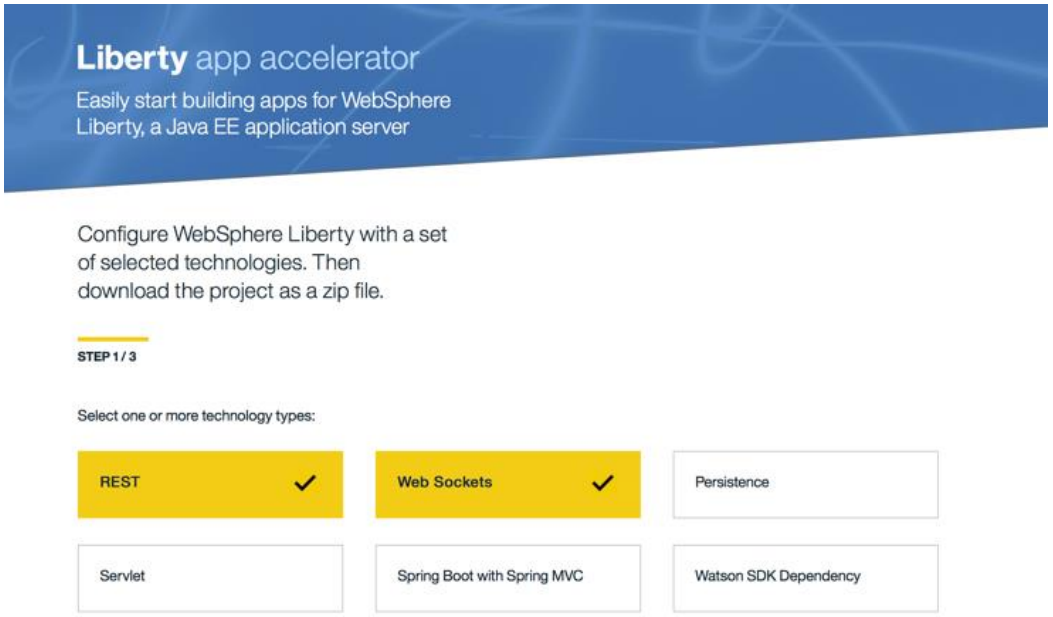
- Dockerコンテナ(ローカル/リモート)で稼働するLibertyの開発環境，デバッグ環境を簡単にセットアップ



※WDT: WebSphere Developer Tools (Eclipseプラグインとして提供する開発ツール)

Liberty app accelerator

- Libertyで動作するサンプルアプリ
  - ランタイム環境の自動構成
    - 指定された API を利用可能な server.xmlの生成、対応APIは順次増加予定
  - コード・スニペットの提供
    - テスト・コードも含めて
  - Maven pom.xml の提供
    - ビルド、デプロイ、テスト



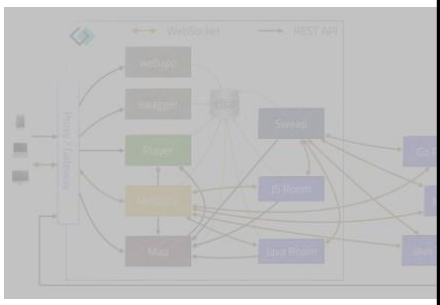
- サンプルの種類

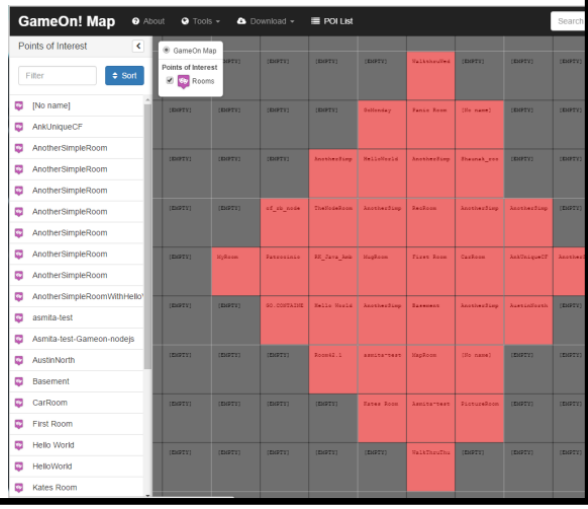
  REST (JAX-RS)
  Web Sockets
  Persistence (JPA)
  Servlet
  Spring Boot with Spring MVC
  Watson SDK Dependency

マイクロサービス・サンプル (OSS提供)

GAMEON A microservices adventure

- Exemplar microservices application
  - Ployglot
    - Core services built using Liberty and Java EE7
    - Deployed using Cloud Foundry, Container, and Event-driven runtimes
  - Parity between development and test
    - Lightweight runtimes and Docker containers support production-like local development
  - Ease of integration and extension
    - Use Java EE7 and Liberty to create extensions for the game
    - JAX-RS for Swagger defined REST APIs
    - WebSockets and Rx-Java for asynchronous messaging.
  - Open source and extensible
    - An “adventure game” where each room is a microservice.
    - Anyone can add a new microservice room to the game instance on Bluemix
    - Rooms scored on availability and adherence to the Room API
    - Rooms with the highest score move towards the middle of the map.





- https://game-on.org/

Connect

Libertyでつなげる，つながるAPI連携

LibertyはAPIレディー

- JAX-RS 2.0 (Java EE 7)をフル活用
- Liberty上のREST APIを公開
- 公開されているAPIをLibertyで活用
- Libertyとクラウドサービスの連携

RESTful Webサービスへの対応

❑Java EE 7で強化されたRESTへの対応

- JSR 339: Java API for RESTful Web Services (JAX-RS) 2.0
  - サーバーAPIだけでなくクライアントAPIも提供
- JSR 353: Java API for JSON Processing (JSONP) 1.0
  - JSON（JavaScript Object Notation）の処理を標準仕様で対応

❑広がるRESTによる連携

- Microservices Architectureでの内部連携
- ブラウザ上のJavaScriptで動くHTML5アプリケーション
- モバイルアプリケーション
- 社外のシステムとのAPI連携
- 社内のデータセンター内のシステム間連携
- クラウド上のアプリとオンプレミス上のシステムとの相互連携

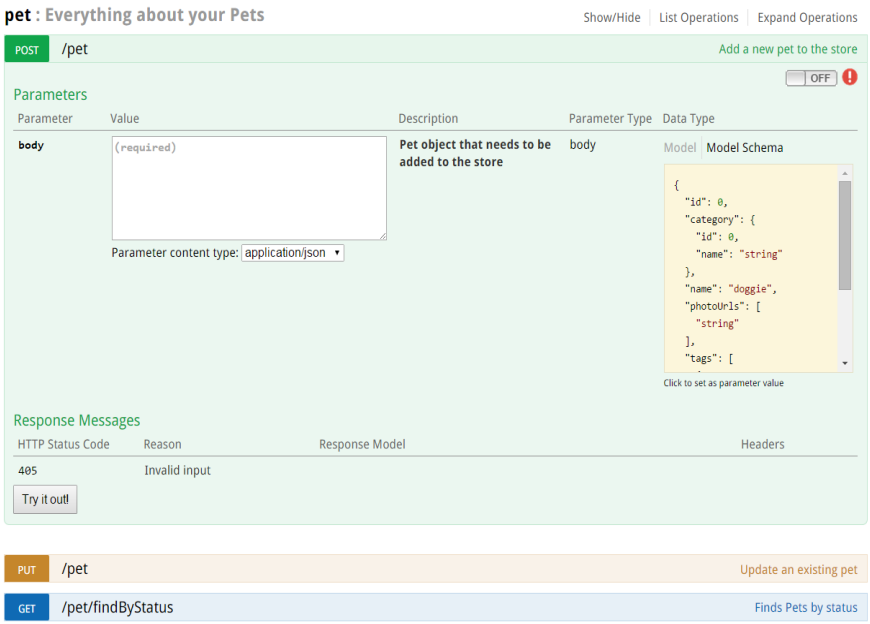
LibertyにおけるAPI の開発方法

- ボトム・アップでの生成
  - コード開発者がAPIのアーキテクチャもデザイン
  - JAX-RS と Swaggerのアノテーションを含む APIコードから生成
- トップ・ダウンでの生成
  - コード実装前にAPIをデザインして開発
  - Swaggerツール を利用しSwagger文書を開発
    - API Connect または 外部ツール
  - 開発された swagger.json を WebモジュールのMETA-INFフォルダに配置
    - APIコードは，アノテーションなしで実装
- 両方
  - APIコード内のアノテーションとMETA-INF内のSwagger文書から生成

Swagger 2.0 http://swagger.io

- REST API の仕様やドキュメントを記述するための標準仕様
  - REST API で使用可能なリソースのリストと，それらのリソースに対する操作を規定
  - パラメーターの名前とタイプ，必須orオプション，取れる値などの規定

- REST API開発に役立つツール群
  - Swagger Editor
    - Swagger 文書の作成を支援するツール
  - Swagger UI
    - Swaggerで定義されたAPIをビジュアルにテスト
  - Swagger CodeGen
    - Swagger文書から様々な言語での呼出し方法の生成



Swagger 2.0 文書の自動生成

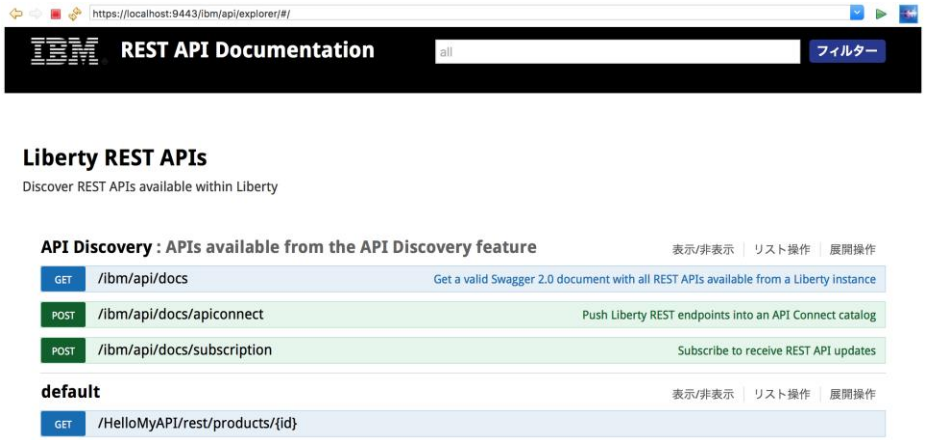
- JAX-RSのクラスの中でSwaggerのアノテーションの埋込み
  - Libertyで <apiDiscovery-1.0> フィーチャーを有効化することで利用可能
  - Swagger文書を自動的に生成

```
…
// アプリケーション・レベル・アノテーション
@SwaggerDefinition(tags={@Tag(name="CloudFirst Bank API", description="APIs for Account Transactions")})
@ApplicationPath(”cfBankAPI")
public class CloudFirstBankApplication extends Application {

  // メソッド・レベル・アノテーション
  @Path("/accounts/{id}")
  @GET
  @Produces("application/json")
  public String getAccounts(@PathParam("id") String id) {
            return MongoDBAccess.getAccounts(id);
  }
}
```

API Discovery

- REST API の活用を促進するための Liberty機能
  - Swaggerで APIドキュメンテーションを公開
    - REST API を公開したいLibertyサーバーで apiDiscovery-1.0を有効化
    - デプロイされたアプリからRESTAPIを集約して提供
      - Swagger.json で指定 or JAX-RSから自動生成
  - アクセスURI
    - JSON or yaml https://<host>:<https_port>/ibm/api/docs
    - Swagger UI https://<host>:<https_port>/ibm/api/explorer
  - Liberty Collective環境
    - Member, Controller双方で有効化することで、MemberのAPI情報もControllerで取得可能

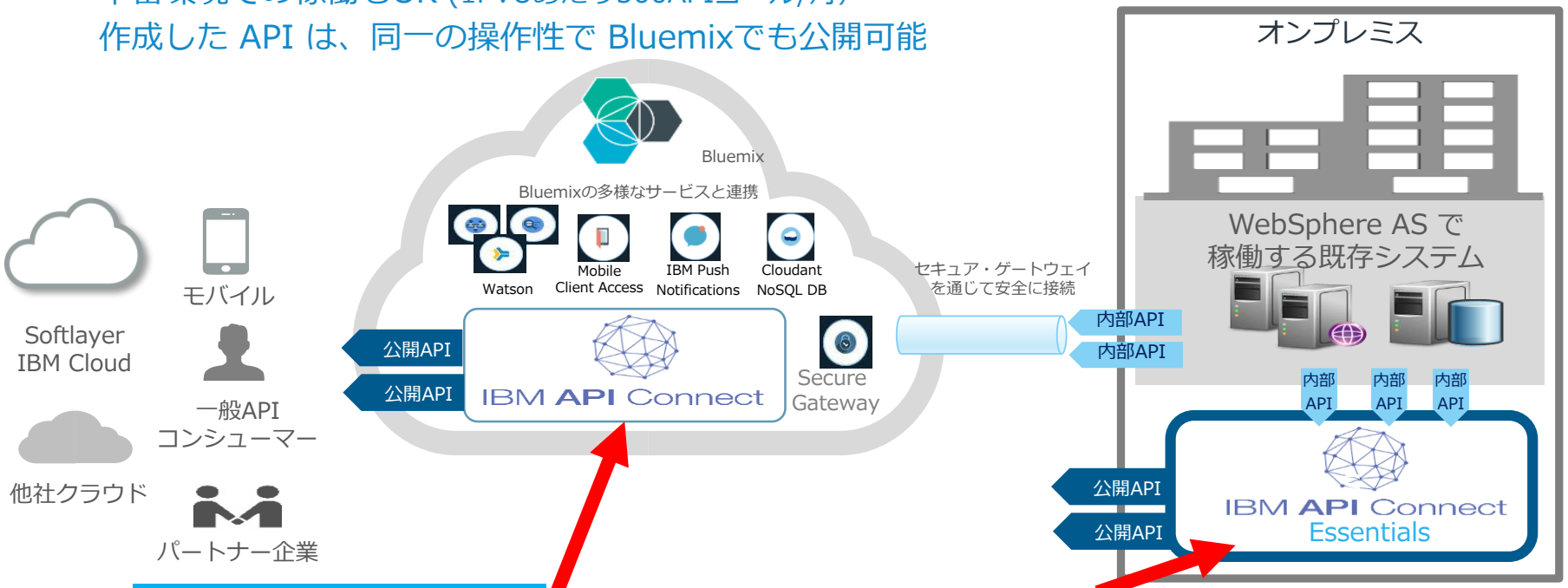


APIを公開

WebSphere Connect

WebSphere Application Serverに **API Connect Essentials** をバンドル

- WebSphere ASの保守契約で、オンプレ上のAPI Connect Essentialsの保守も提供
- 本番環境での稼働もOK (1PVUあたり500APIコール/月)
- 作成した API は、同一の操作性で Bluemixでも公開可能

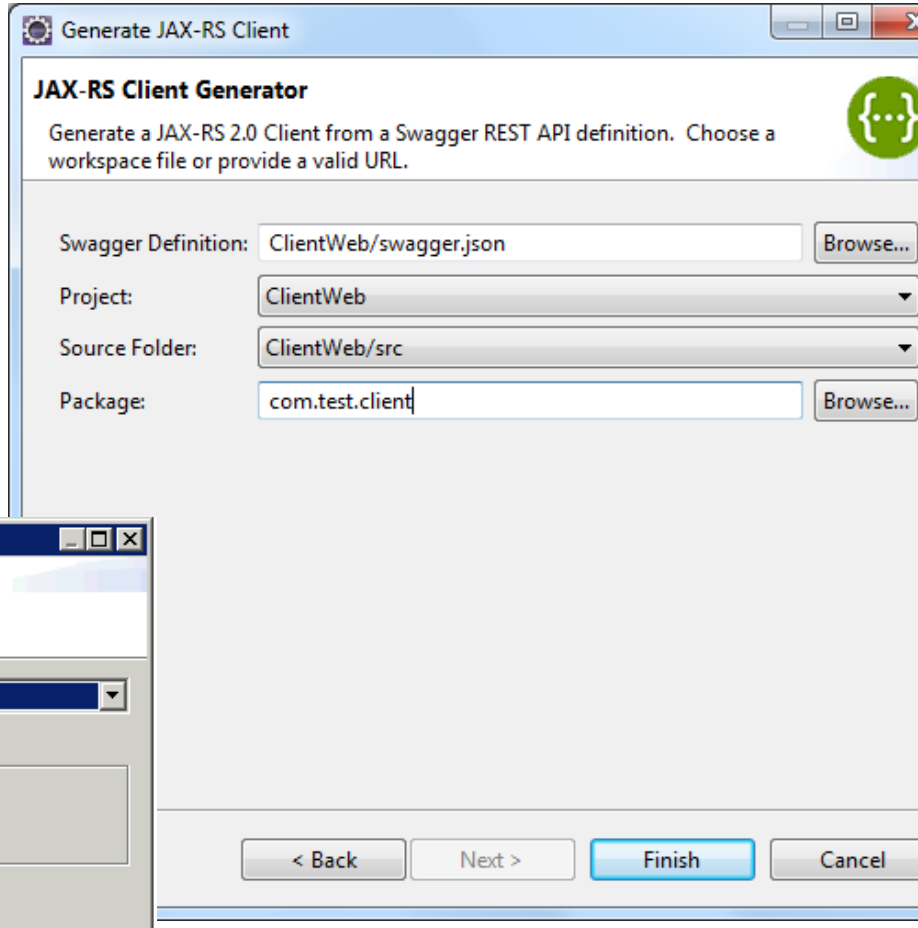


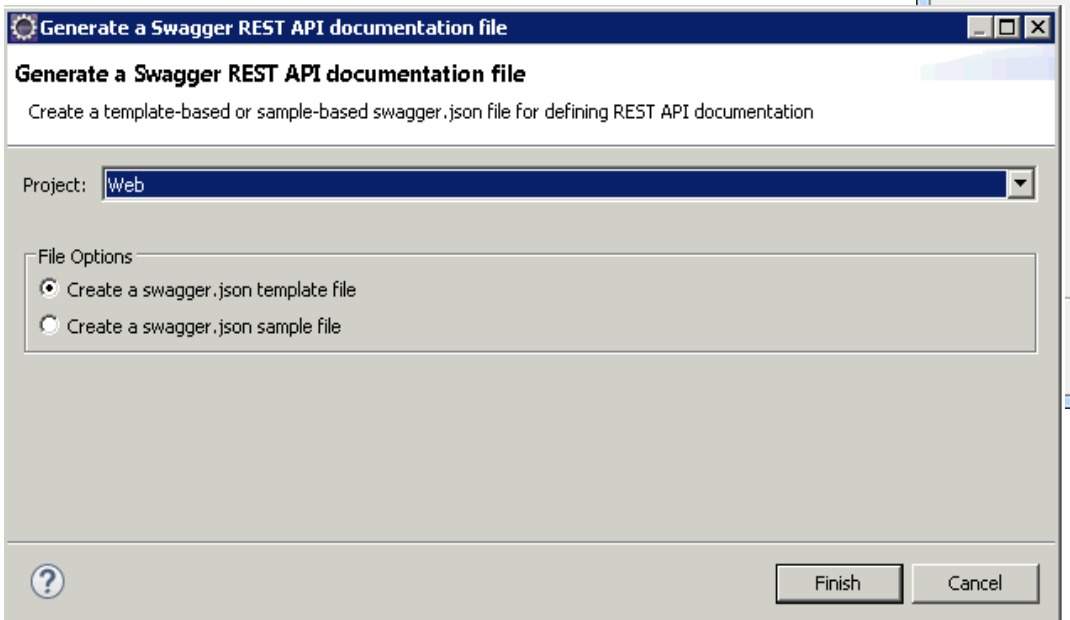
IBM API Connect

Liberty から自動生成したSwaggerをインポートして，APIを公開するソリューション

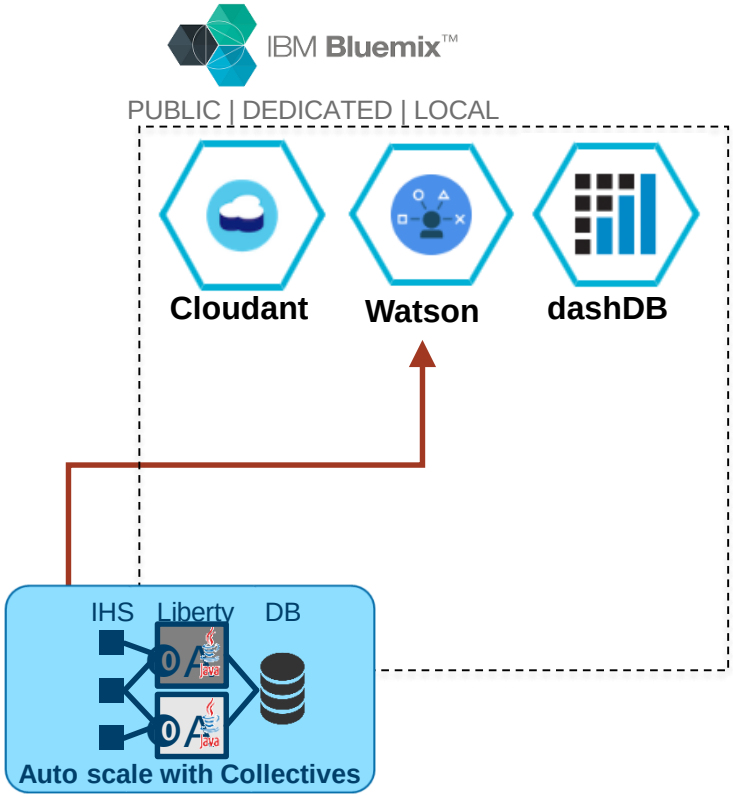
WDTのAPI機能強化

- Swaggerドキュメントの生成
  - サンプル or テンプレート
- JAX-RSクライアントの生成
  - Swaggerドキュメントから
    クライアントコードを自動生成





Bluemix Utility



- bluemixUtilityコマンド
  Bluemixサービスを検索し，インポートするためのコマンド
  bluemixUtility import <serviceName>

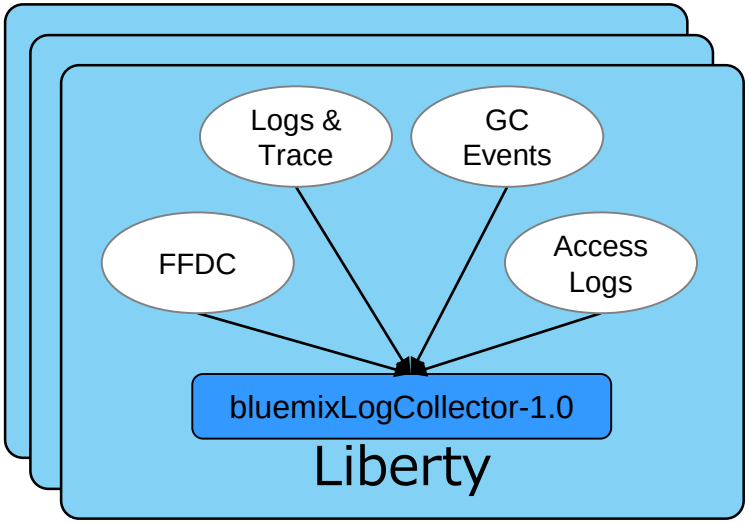
- クライアントライブラリのダウンロード
- サーバー構成へのバインディング

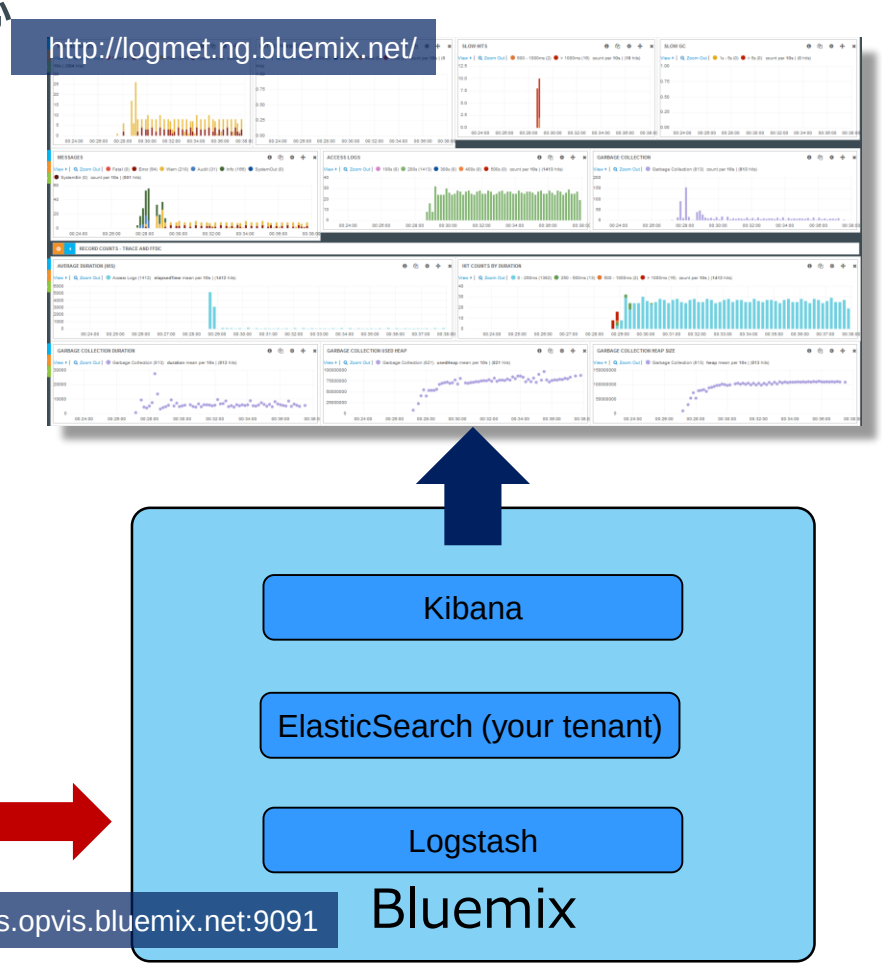
```
<jndiEntry jndiName="watson/text_to_speech/apiKey"
           value="NzNmNT"/>
```

- 接続先のサービス
  - Cloudant
  - Watson
  - dashDB

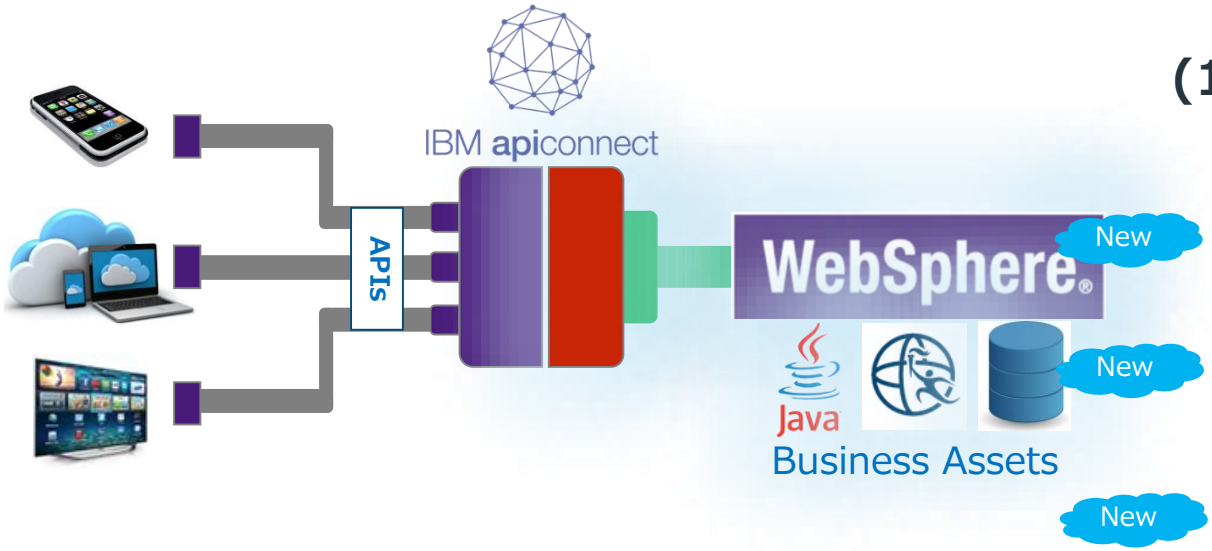
Log Analytics with Bluemix

- ログ閲覧が可能なダッシュボード (Bluemix上)
  - 複数Libertyサーバーに対応
- ELKベース
  - ElasticSearch
  - Logstash
  - Kibana





Liberty上のREST APIを**公開する**



(1) Libertyアプリを REST APIで公開

- JAX-RS または Servlets を APIs 化
- Swagger ドキュメント化 (annotations or separate doc)
- apiDiscovery-1.0 フィーチャー で /ibm/api エンドポイントで公開
- API Connectと連携

WebSphereアセットを API で公開すると…

- 既存のWASアプリケーションのAPIを社内，社外の新しいアプリケーションに公開することで，**追加の価値を創出**
- APIの作成に，**すでに持っているWebSphereとJavaのスキルを活用**
- 提供する製品やサービスに**新しい顧客をひきつけ**，APIエコシステムを構築し，新しい売上の源泉を創出

Libertyでクラウドサービス**や**API**を使う**

(2) Liberty からBluemixクラウドサービスを呼び出す

- New bluemixUtilityで容易に連携
- Watson, Cloudant, etc.
- New API ConnectからSwaggerインポート
- ログ収集，ビジュアライゼーション
- 今後も連携するサービスを追加予定

革新的なBluemixクラウドサービスを呼び出し、**Liberty アプリケーションを拡張**

これらのクラウドサービスは，**低コスト，低複雑性，低リスク**で管理される

Liberty の機能を活用することで，LibertyからシームレスにBluemixクラウドサービスを利用できる

Optimize

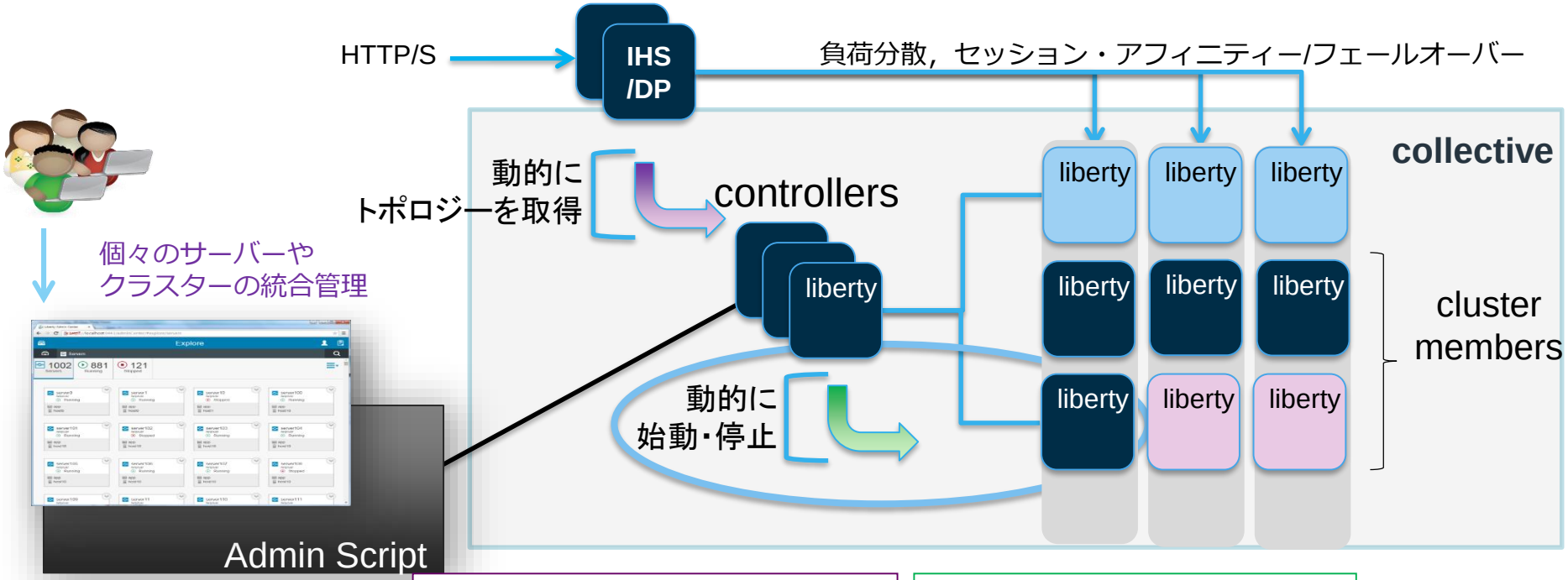
Libertyでプロダクション・レディー

Libertyはプロダクション・レディー

- Intelligent Managementの機能強化
- CollectiveのPolyglot対応
- traditional WASに引けをとらないパフォーマンス

Intelligent Managementの機能強化

Collectives



動的ルーティング
・サーバー・アプリの起動/停止/追加/削除を認識
・plugin-cfg.xmlの編集不要

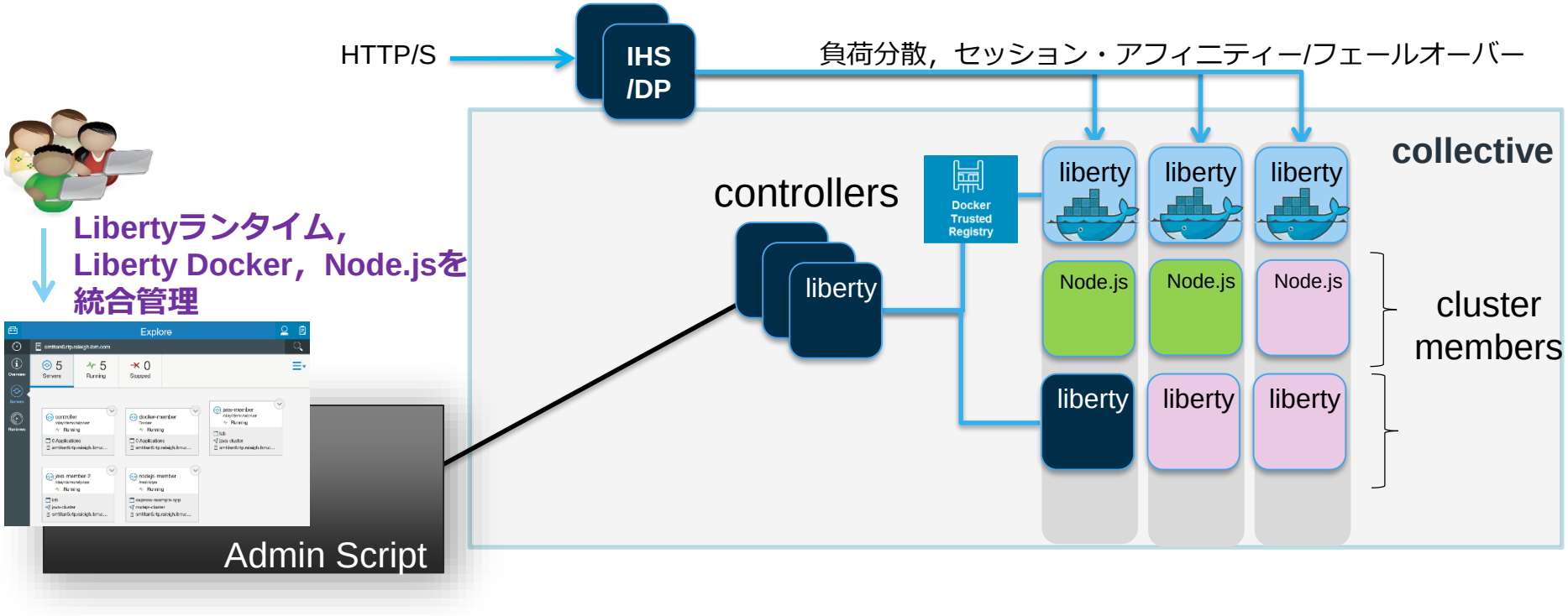
自動スケーリング
・ポリシーベース(リソース閾値，最大・最小インスタンス数)

ヘルス管理
・監視条件の設定，自動アクション

保守モード
・サービスに影響を与えずに切離し

Libertyに加えてNode.js，Dockerもサポート

Collectives

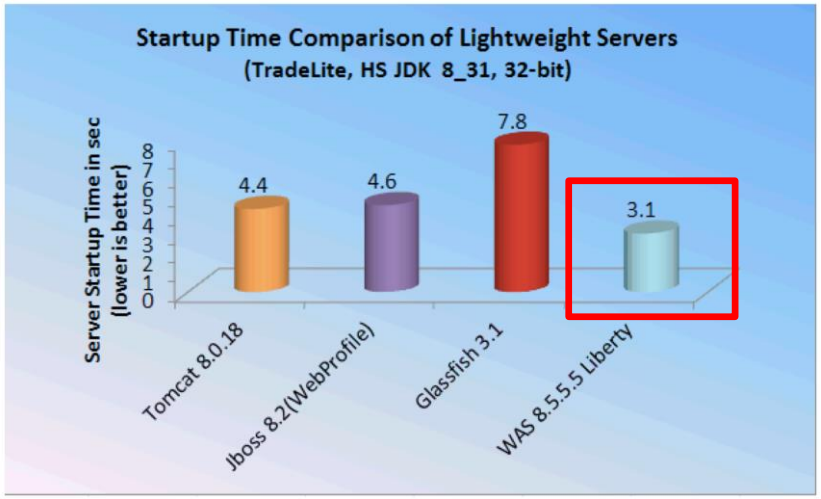


集合を使用した、Liberty 用のサーバー管理環境のセットアップ
https://knowledgecenters.hursley.ibm.com/liberty-refresh/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/tagt_wlp_server_management.html?cp=SSAW57_liberty%2F3-1&lang=ja

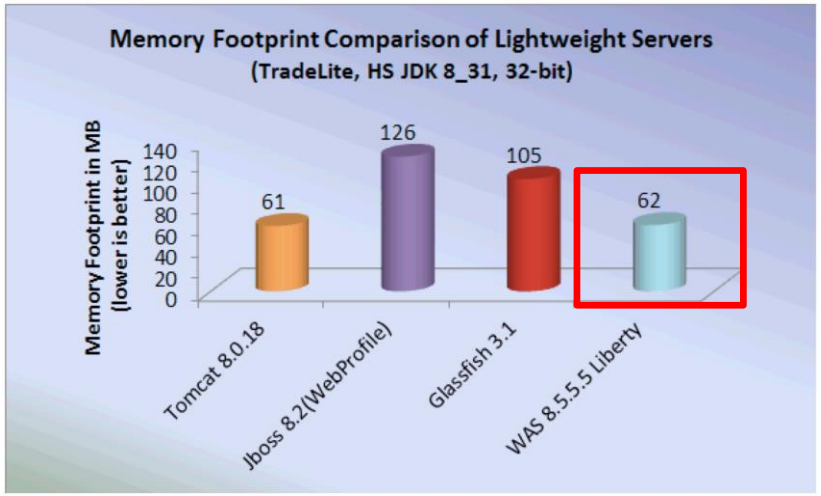
デプロイメント REST API を使用した Node.js サーバーのデプロイ
https://knowledgecenters.hursley.ibm.com/liberty-refresh/api/content/nl/ja/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_deployservice_nodejs.html

デプロイメント REST API を使用した Liberty Docker コンテナーのデプロイ
https://knowledgecenters.hursley.ibm.com/liberty-refresh/api/content/nl/ja/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_deployservice_docker.html

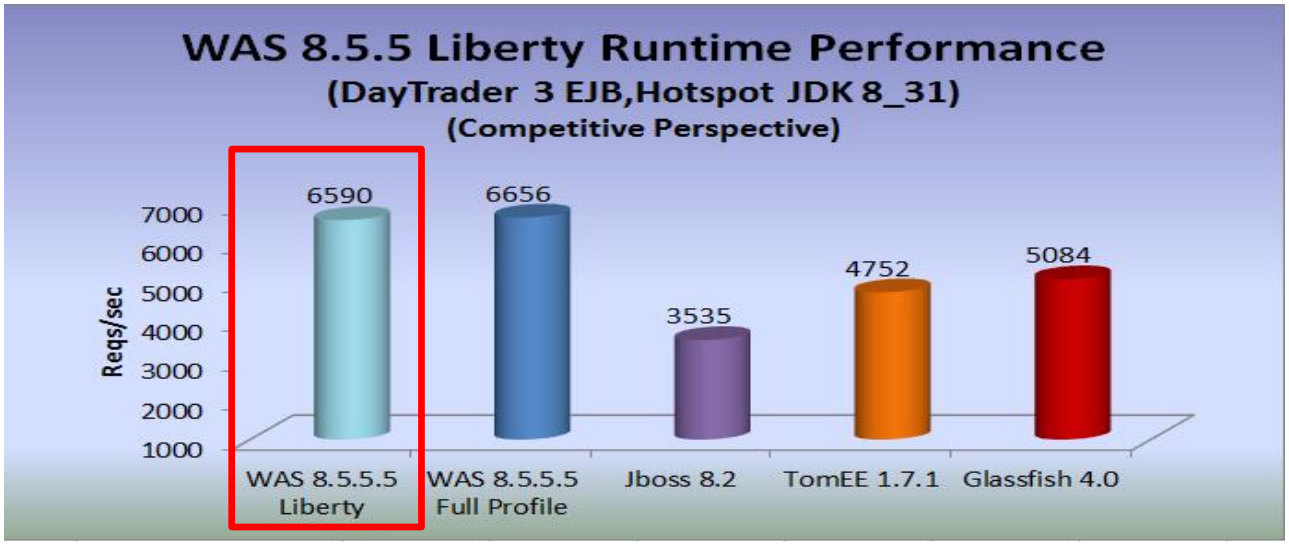
パフォーマンス性能



高速なサーバー起動



少ないメモリ消費量



高いパフォーマンス

まとめ

- WAS Liberty特長
  - 軽量・高速
  - Java EE 7 + Liberty独自フィーチャー
  - ゼロ・マイグレーション
- Create
  - クラウド・ネイティブを強力支援
- Connect
  - API を公開する/利用するための機能強化
- Optimize
  - プロダクション・レディー

WebSphere Application Server

以下，補足

- WASとは
- WAS Libertyの主要エディション
- WAS Libertyの特長
  1. Java EE 7対応
  2. 軽量・高速起動
  3. Unzipで簡単パッケージング，デプロイ
  4. シンプルで簡単なサーバー構成ファイル
  5. 便利な開発ツール
  6. DevOpsに最適な各OSSツール連携
  7. クラウド対応強化
  8. API機能強化

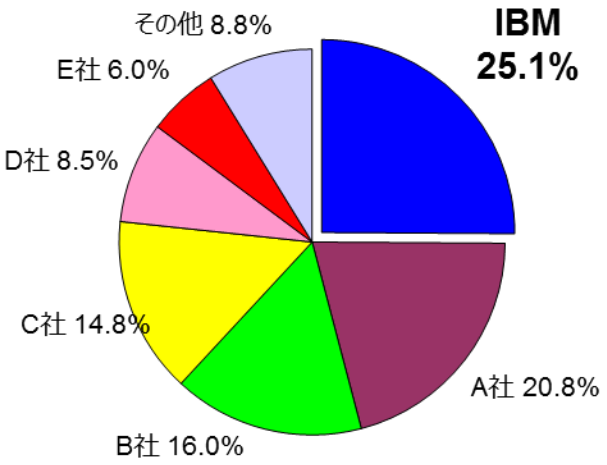
WebSphere Application Serverとは

- WebSphere Application Server (WAS)は,
  Java EE 仕様に従って作成されたエンタープライズ・アプリケーションを実行するプラットフォーム

  - 1998年より継続的に提供
  - 2016年6月 V9 リリース(※1)
  - 7年連続国内シェア No.1

●業界標準技術への対応
●信頼性・管理機能の強化
●製品戦略に基づく一貫した機能拡張

デプロイメントセントリック
アプリケーションプラットフォーム市場

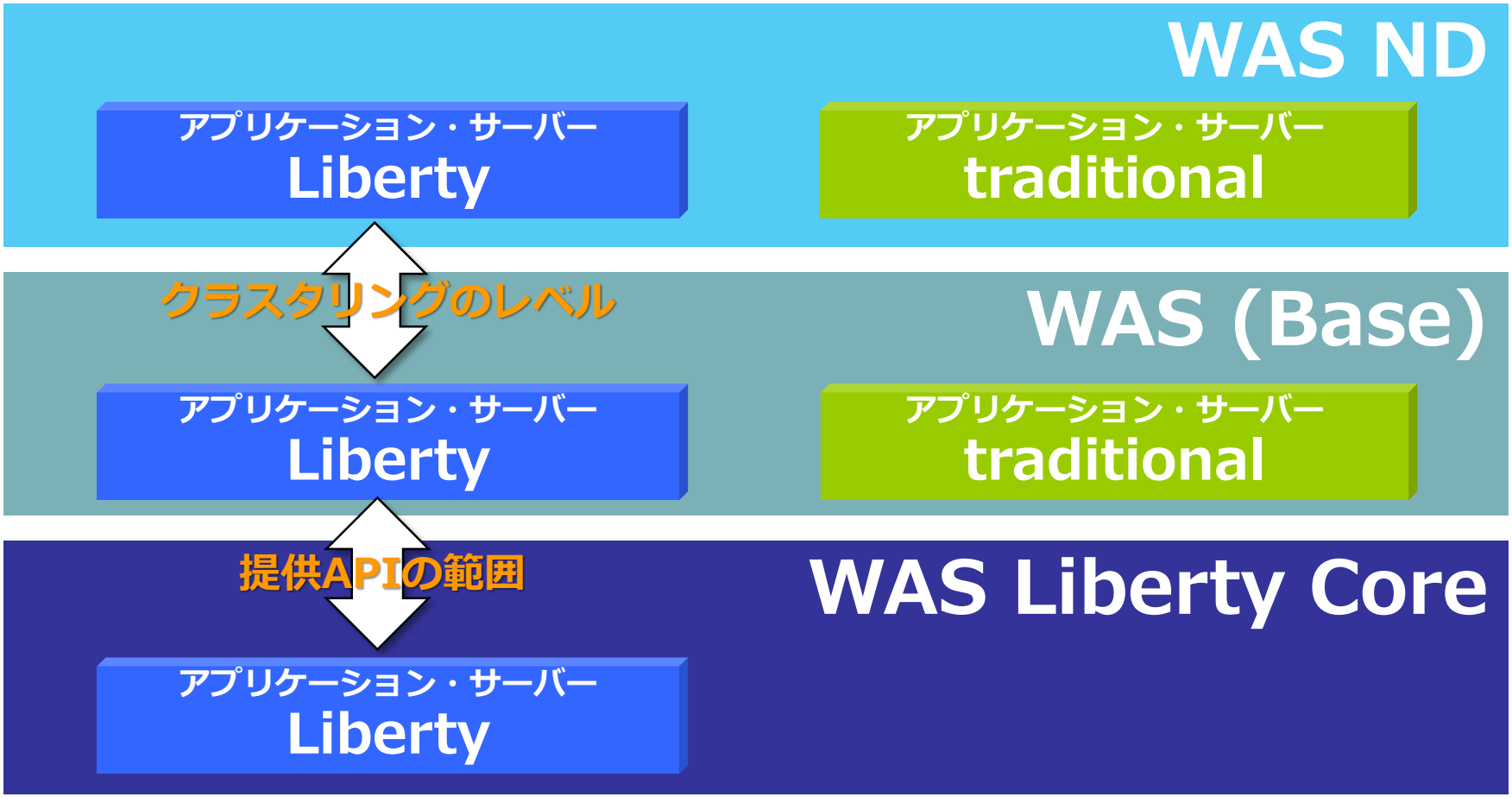


IDC Japan　2015年8月

出典：IDC #J15380106 「国内アプリケーションデプロイメント／構造化データ管理ソフトウェア市場2014 年の分析と2015 年～2019年の予測（2015年8月）

WAS Libertyの主要なエディション

WAS Libertyは全てのエディションで使用可能なランタイムです。
エディションによって，機能差があります。



WAS Libertyの特長

①Java EE 7対応
Java EE 7 標準に準拠したアプリを完全サポート
JAX-WS, JAX-RS, JMSもサポート
新機能も継続的に提供

②軽量ランタイム
メモリー使用量が小: 60MB程度
ディスク使用量も100MB以下
起動が速い：5秒程度

③Unzipによる導入とデプロイ
パッケージをした
サーバー ＋ アプリ ＋ 構成情報を
Unzipでデプロイ可能

④簡単な構成と動的変更
最低限必要な構成ファイルはserver.xmlひとつだけ
デフォルトベースで簡単構成
構成変更は再起動なしに反映

WAS Liberty & WDT

⑤統合ツール（WDT)
高機能なEclipse用の連携ツール
を無償で提供
Eclipseから簡単に使用可能

⑥自動化ツールとの連携
多くのOSSツールに
無償でプラグインを提供

⑦様々な環境で稼動
オンプレ，クラウド(IaaS、
PaaS)，Dockerで稼動

⑧API公開
RESTのアノテーションから
Swaggerを自動生成

1. Java EE 7 完全対応

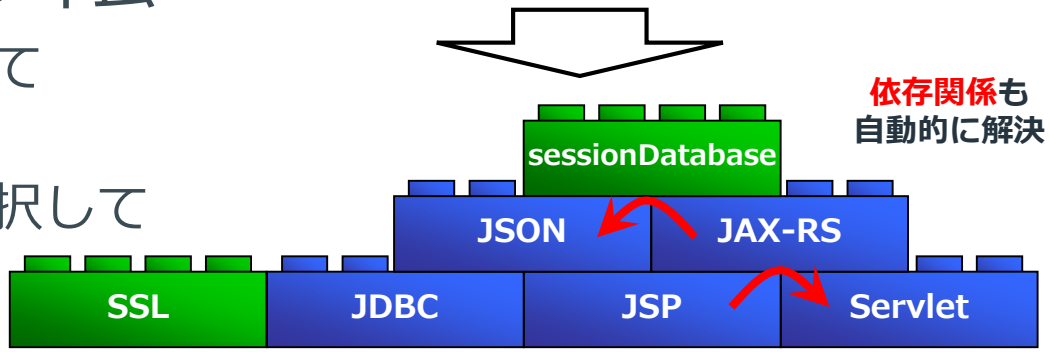
- JSR 342: Java Platform, Enterprise Edition 7 の主なAPI群
  - HTML5環境への対応
    - JSR 344: JavaServer Faces (JSF) 2.2
    - JSR 353: Java API for JSON Processing (JSONP) **1.0**
    - JSR 356: Java API for WebSocket **1.0**/1.1
    - JSR 339: Java API for RESTful Web Services (JAX-RS) 2.0
  - 開発生産性の向上
    - JSR 345: Enterprise JavaBeans (EJB) 3.2
    - JSR 346: Contexts and Dependency Injection for Java EE (CDI) 1.1/1.2
    - JSR 907: Java Transaction API (JTA) 1.2
    - JSR 349: Bean Validation 1.1
  - エンタープライズ・ニーズへの対応
    - JSR 343: Java Message Service (JMS) 2.0
    - JSR 338: Java Persistence API (JPA) 2.1
    - JSR 236: Concurrency Utilities for Java EE **1.0**
    - JSR 352: Batch Applications for the Java Platform **1.0**

2. 軽量・高速なランタイム

- 軽量
  - 数十メガバイトのメモリ消費／100メガバイト程度のディスク消費
  - コンテナや仮想環境への集約が容易に
- 高速起動・動的変更
  - 数秒以内でサーバーが起動
  - サーバーの構成変更やアプリケーションの変更も即座に反映
- モジュール構造のランタイム
  - 提供機能をFeatureとしてモジュール化
  - 必要なFeatureだけを選択して導入・起動

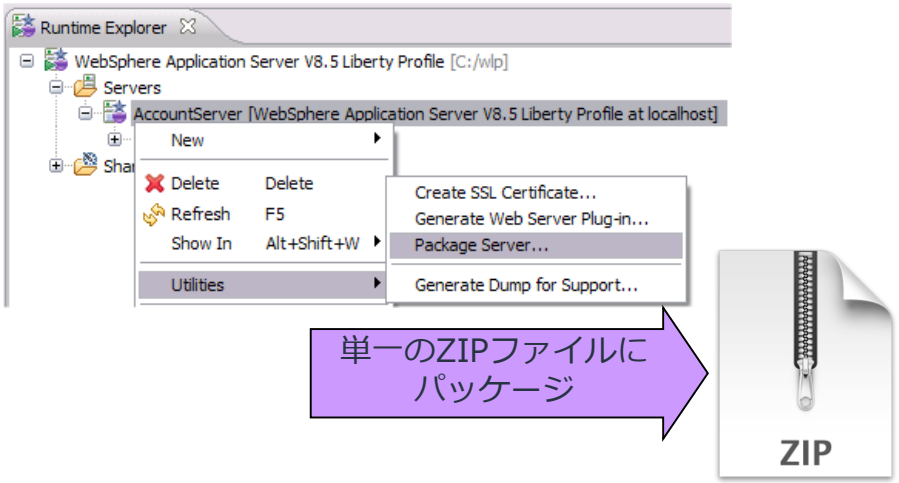
構成ファイル server.xml

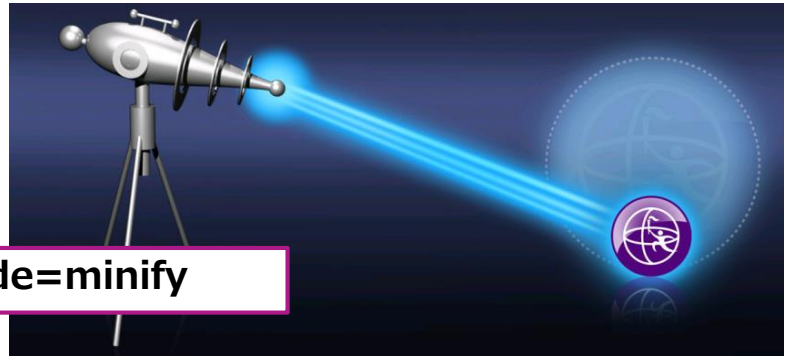
```
<featureManager>
    <feature>jsp-2.3</feature>
    <feature>jdbc-4.1</feature>
    <feature>jaxrs-2.0</feature>
    <feature>sessionDatabase-1.0</feature>
    <feature>ssl-1.0</feature>
</featureManager>
```



3. ZIP展開による導入

- 開発・テストした環境を ZIPにそのままパッケージ化
  - サーバー環境・構成
  - アプリケーション・リソース
- ZIPを展開するだけで導入
  - インストーラーの実行や OSの構成変更などは不要
- アプリケーションが使ってる フィーチャーのみを パッケージすることも





```
$ server package serverName --include=minify
```

4. シンプルで柔軟な構成

- 基本はserver.xmlという単一のファイル
- 全ての構成項目がデフォルトの値をもつ
  - デフォルトから変更する場合にのみ記述
  - 簡潔な構成ファイル

バージョン
管理システムでの
履歴管理も
容易

traditional WASの構成ファイルの例

```
<?xml version="1.0" encoding="UTF-8"?>
<security:Security xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:orb.securityprotocol="http://www.ibm.com/websphere/appserver/schemas/5.0/orb.securityprotocol.xmi"
xmlns:security="http://www.ibm.com/websphere/appserver/schemas/5.0/security.xmi" xmi:id="Security_1"
useLocalSecurityServer="true" useDomainQualifiedUserNames="false" enabled="true" cacheTimeout="600"
issuePermissionWarning="true" activeProtocol="BOTH" enforceJava2Security="false"
enforceFineGrainedJCASecurity="false" appEnabled="false" dynamicallyUpdateSSLConfig="true"
allowBasicAuth="true" activeAuthMechanism="LTPA_1" activeUserRegistry="WIMUserRegistry_1"
defaultSSLSettings="SSLConfig_1">
<authMechanisms xmi:type="security:KRB5" xmi:id="KRB5_1" OID="oid:1.2.840.113554.1.2.2"
authContextImplClass="com.ibm.ISecurityLocalObjectTokenBaseImpl.Krb5WSSecurityContextImpl"
authConfig="system.KRB5" simpleAuthConfig="system.KRB5" authValidationConfig="system.KRB5"
krb5Realm="" krb5Config="" krb5Keytab="" krb5Spn="WAS/${HOST}" trimUserName="true"
enabledGssCredDelegate="true"/>
<authMechanisms xmi:type="security:LTPA" xmi:id="LTPA_1" OID="oid:1.3.18.0.2.30.2"
authContextImplClass="com.ibm.ISecurityLocalObjectTokenBaseImpl.WSSecurityContextLTPAImpl"
authConfig="system.LTPA" simpleAuthConfig="system.LTPA" authValidationConfig="system.LTPA"
timeout="120" keySetGroup="KeySetGroup_1">
<trustAssociation xmi:id="TrustAssociation_1" enabled="false">
<interceptors xmi:id="TAInterceptor_1"
interceptorClassName="com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus"/>
<interceptors xmi:id="TAInterceptor_2"
interceptorClassName="com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl"/>
</trustAssociation>
<singleSignon xmi:id="SingleSignon_1" requiresSSL="false" domainName="" enabled="true"/>
</authMechanisms>
<authMechanisms xmi:type="security:SPNEGO" xmi:id="SPNEGO_1" OID="oid:1.3.6.1.5.5.2"
authContextImplClass="com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl"
allowAppAuthMethodFallback="false"/>
```

WAS Libertyの最小構成ファイルの例

```
<server description="new server">
    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.3</feature>
    </featureManager>
</server>
```

5. WDT (WebSphere開発ツール)

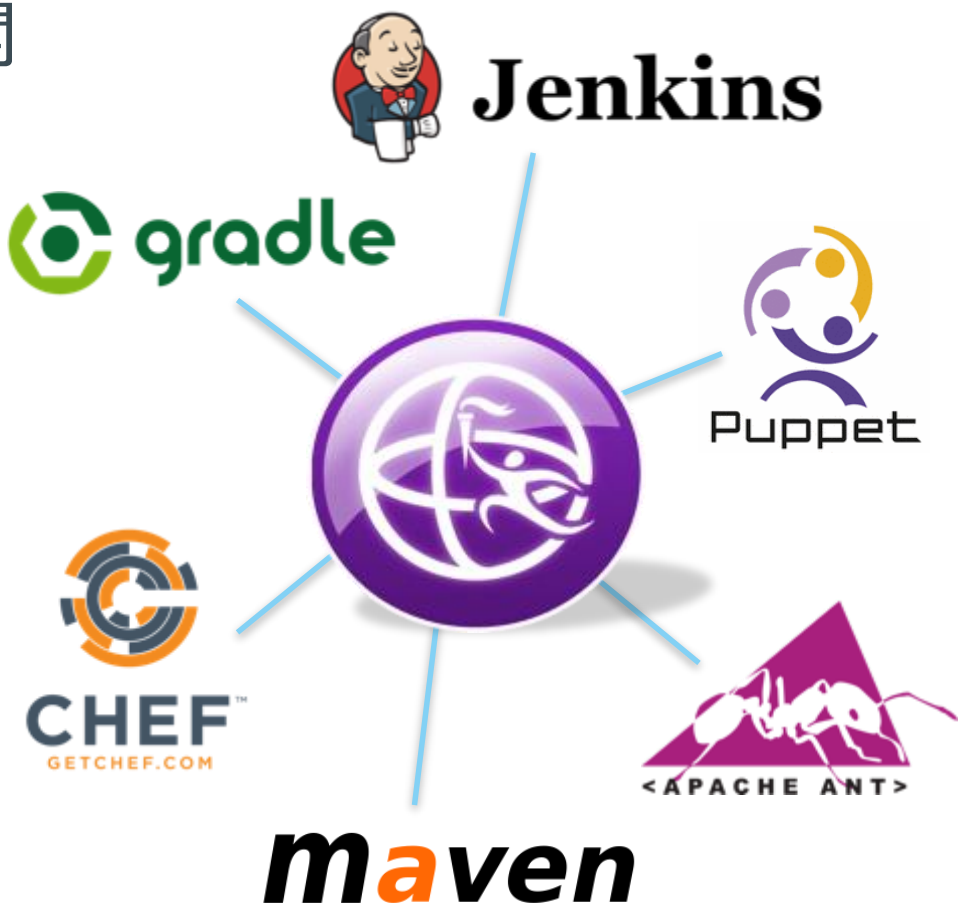
- WDT (WAS Developer Tools)
  - 無償利用可能な開発ツール
  - Eclipseプラグインとして導入可能

  - Libertyのサーバー構成ファイルの編集に最適
    - server.xmlをコーディングレスで編集可能

  - 連携 (デプロイや始動・停止の制御)
    - リモートのLiberty環境と連携
    - Bluemixと連携
    - Dockerと連携
    - API Discovery機能によるSwagger自動生成

6. 各種OSSツールとの連携

- 各種CI・Buildツールとの連携機能をGitHubで公開



https://github.com/wasdev

場所を選ばず
あらゆる環境で稼働

7. Libertyは様々なクラウド環境で稼働

オンプレミス

全体の管理
柔軟性の最大化

Build Your Own Cloud

既存のH/W上に、仮想化されたWebSphere Application Serverを使用可能

PureApplication System & Software

再利用 & 再デプロイ可能なパターンとして WebSphere App Server が使用可能

docker

IaaS

パブリッククラウド
Time to Market

SoftLayer/Bluemix

BYOS&L - WebSphere App Server

SOFTLAYER

PureApplication Service
SOFTLAYER
an IBM Company

Amazon

BYOS&L - WebSphere App Server

amazon web services™

Microsoft Azure

BYOS&L - WebSphere App Server
Pay-as-you-Go WAS VMs

Microsoft Azure

docker

PaaS

パッケージ・サービス
高速開発

Bluemix

組立可能なサービス
Liberty Buildpack

Cloud Foundry

Liberty Buildpack

CLOUD FOUNDRY™

docker

8. API機能強化

- API公開のためのSwagger文書の作成
  - JAX-RSのリソースクラスからSwagger文書を自動生成
  - WDT(開発ツール)でテンプレのSwagger生成

- API利用のためのクライアントの構成
  - SwaggerインポートによるJAX-RSクライアントの生成

- クラウドサービスとの連携
  - Bluemix Utilityによる接続(Watson/Cloudant/dashDB/etc.)
  - 複数のLibertyサーバーのログ閲覧のためのBluemix連携