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

WAS V9 traditional (フルプロファイル)

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
中島 由貴



アジェンダ

- WAS V9 traditional 新機能ハイライト
- Java EE 7
- Java SE 8
- tWAS クラウド対応
- V8.0からのアップデートとトポロジー

WAS V9 traditional 新機能ハイライト

traditional WAS (tWAS) 新機能ハイライト(1)

- Java EE 7 対応
  - Libertyですでに実績がある共通コード

- Java SE 8 対応
  - IBM Java SDK 8 が同梱かつ唯一の前提Java
    - (補足) tWAS 8.5.5.9 (2016年3月提供) / Liberty 8.5.5.5 (2015年3月提供) でJava SE 8を追加サポート

新Java EE/SE仕様対応以外(構成管理など)は、
V8.5.5から大きな変更なし
信頼と実績のWAS V9 traditional

traditional WAS (tWAS) 新機能ハイライト(2)

- API 対応
  - API Connect Essentialsを同梱
    - WASのバージョンに関わらず、利用可能

- クラウド対応
  - WAS for Bluemix
    - tWAS V9 も提供
    - シングル・テナント・オプションも提供予定
  - tWAS Dockerイメージ提供

Java EE 7

Java EE 7 の3つのゴール

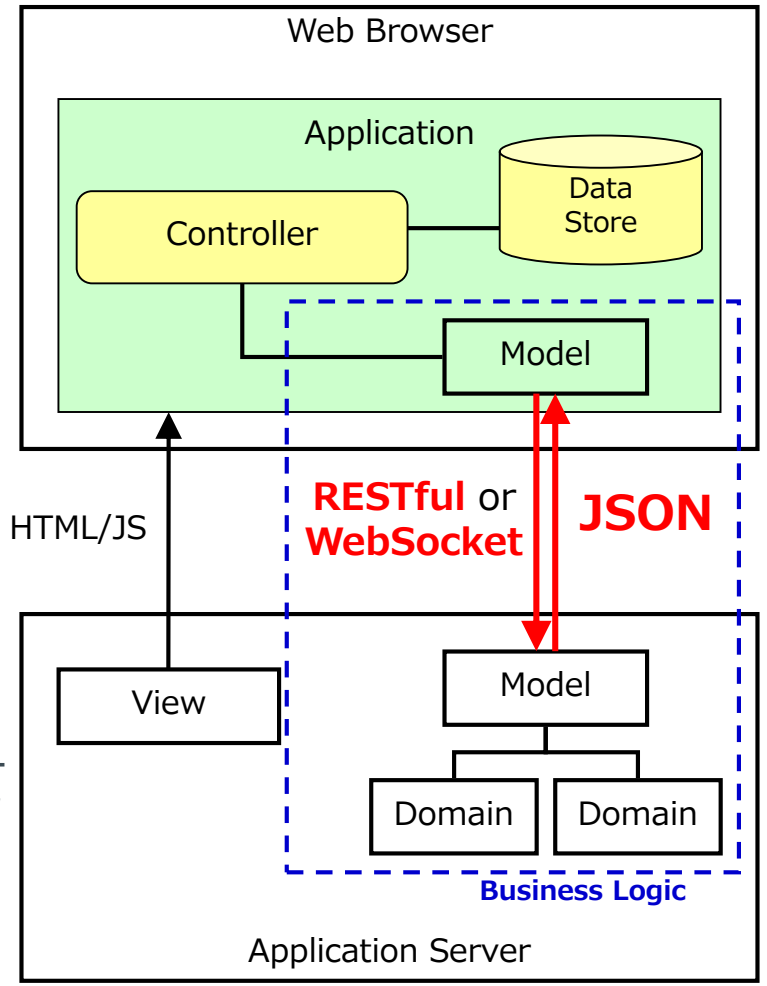
- HTML5環境への対応
- 開発生産性の向上
- エンタープライズ・ニーズへの対応

Java EE 7 の主なAPI群

- JSR 342: Java Platform, Enterprise Edition 7
  - 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**

HTML5におけるクライアント・サーバー連携

- 通信方法
  - RESTful
    - HTTPの原則にしたがった簡潔なシステム間連携の設計手法
  - WebSocket
    - ブラウザとWebサーバーの間で双方向通信を可能にする通信プロトコル
- データ型
  - JSON（JavaScript Object Notation）
    - JavaScriptのオブジェクトとしてそのままパースできるテキスト形式のデータ表記法



WebSocketとは

- クライアントとサーバー間の双方向通信用の技術規格
  - TCP上で動く
  - HTTP UpgradeリクエストによりWebSocketにプロトコルスイッチする
- 仕様
  - WebSocket Protocol
    - 元々はHTML5の仕様の一部として策定されていた
    - その後HTML5から切り離され、単独のプロトコルとしてIETF RFC 6455で仕様化
  - WebSocket API
    - JavaScript APIはW3Cで仕様化
  - Java API for WebSocket 1.0はJava EE 7(JSR 356) で仕様化
- 用途
  - リアルタイムアプリケーション
    - Single Page Applicationとして画面遷移することなく、画面表示をリアルタイムに更新できる
    - 例）オンライントレード、モニタリング、オンラインゲーム、チャット、SNS、画面共有など

WebSocketサーバーエンドポイントの実装

- サーバーエンドポイント
  - サーバー側はJava EE(@ServerEndpoint)を使用する

```
@ServerEndpoint("/hello")
public class HelloEndpoint{
    Session currentSession = null;
    //接続がオープンしたとき
    @OnOpen
    public void onOpen(Session session, EndpointConfig ec) {
        currentSession = session;
    }
     //メッセージを受信したとき
    @OnMessage
    public void receiveMessage(String msg) throws IOException {
        //メッセージをクライアントに送信する
        currentSession.getBasicRemote().sendText("Hello" + msg);
    }
    //接続がクローズしたとき
    @OnClose
    public void onClose(Session session, CloseReason reason) { ・・・ }
    //接続エラーが発生したとき
    @OnError
    public void onError(Throwable t) {・・・ }
}
```

サーバーからWebSocketメッセージの送信

１つのエンドポイントに対してメッセージを送信するステップは以下の通り

1. connectionからSessionオブジェクトを取得する
   - ピアからのメッセージに対する応答を送信する場合は@OnMessageアノテーションが付加されたメソッド内でSessionオブジェクトが利用できる

     ```
     @OnMessage
     public void message(Session session, String msg) {}
     ```

   - 送信メッセージがピアへの応答ではない場合、@OnOpenアノテーションが付加されたメソッド内でsessionオブジェクトをインスタンス変数としてストアしておく

     ```
     @OnOpen
     public void onOpen(Session session, EndpointConfig ec) {
           currentSession = session;
     }
     ```

2. SessionオブジェクトからRemoteEndpointオブジェクトを取得する
   - Session.getBasicRemoteメソッドでRemoteEndpoint.Basicオブジェクトが返される(同期)
   - Session.getAsyncRemoteメソッドでRemoteEndpoint.Asyncオブジェクトが返される(非同期)

3. RemoteEndpointオブジェクトを使用してピアに対してメッセージを送信する
   - テキストメッセージを送信する場合

     ```
     void RemoteEndpoint.Basic.sendText(String text)
     ```

   - バイナリメッセージを送信場合

     ```
     void RemoteEndpoint.Basic.sendBinary(ByteBuffer data)
     ```

   - pingフレームを送信する場合

     ```
     void RemoteEndpoint.sendPing(ByteBuffer appData)
     ```

   - pongフレームを送信する場合

     ```
     void RemoteEndpoint.sendPong(ByteBuffer appData)
     ```

JAX-RS 2.0 と JSONP 1.0

- JAX-RS 2.0の新機能
  - JAX-RS クライアント機能の追加
    - クライアント側でも非同期処理をサポート
  - 非同期処理のサポート
  - Bean Validation 対応
  - プロバイダー機能の拡充
    - エンティティー・インターセプターの追加
    - フィルター機能の追加
- JSONP (Java API for JSON Processing)
  - JSONデータの解析、生成、変換、照会を行うためのAPIを提供

JAX-RSでのJSONPオブジェクトとのマッピング

- JSONP が利用可能な環境では、以下のマッピングをサポート
  - JSONP : JSR 353: Java API for JSON Processing

Java の型	メディア・タイプ
javax.json.JsonStructure, javax.json.JsonObject , javax.json.JsonArray	application/json

- リソース・メソッドでの利用例

```
@Path("{id:[A-Z][0-9]+}")
@GET
public Response queryEmp( … ) {
    // 指定された社員の情報を返す
     … … …
    JsonObjectBuilder ob = Json.createObjectBuilder();
    ob.add( "name",     name );
    ob.add( "ctryCode", ctry );
     … … …
    JsonObject obj = ob.build();
    return Response.ok( obj ).build();
}
```

```
@Path("{id:[A-Z][0-9]+}")
@PUT
public Response updateEmp( JsonObject obj ) {
    // 指定された社員の情報を更新する
    String name = obj.getString( "name" );
    String ctry = obj.getString( "ctryCode" );
     … … …
    return null;
}
```

JAX-RS クライアント

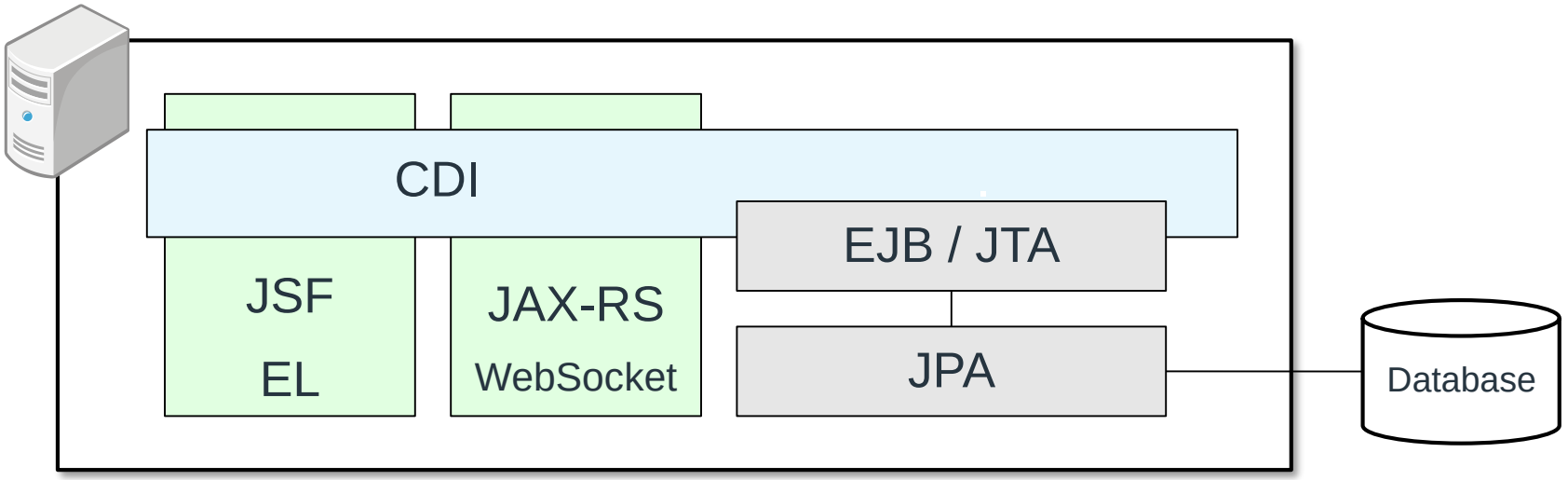
- JAX-RS 2.0 から JAX-RS クライアントの機能を提供
  - javax.ws.rs.client パッケージ
  - 以下のステップで REST サービスを呼び出す
    (1) Client のインスタンスを取得
    (2) WebTarget を作成
    (3) リクエストを生成
    (4) サービスを呼び出す (リクエストを実行/サブミットする)

```
Client client = ClientBuilder.newClient();
Response res  = client.target( "http://www.example.org/contextRoot/appPath/employee" )
                .queryParam( "country", "jp" )
                .request( "application/json" )
                .header( "Range", "items=0-24" )
                .get();
```

メソッド呼び出しを繋げることで、準備から実装までが行える -- "fluent API"

Java EE 7の標準的なアプリケーション構成

- 画面デザイン
  - JSF / EL / Servlet / JSP
- 外部連携
  - JAX-RS / WebSocket
  - JAX-WS
- 依存性注入・ビジネスロジック
  - CDI / EJB / JTA
- DB連携・ORマッピング
  - JPA



Java EE 7 - 登場の意味

2000年代前半のWebアプリケーション

- まだまだ未完成で力不足のJ2EE仕様
  →多くの「アンチJ2EE技術」の登場
- コンテナ上にフレームワークを追加し
  その上にアプリケーションを構築することが主流に
  - Open Sourceフレームワーク
  - ベンダー製フレームワーク
  - 独自フレームワーク

Apache Log4J

Struts

tapestry

iBATIS

HIBERNATE

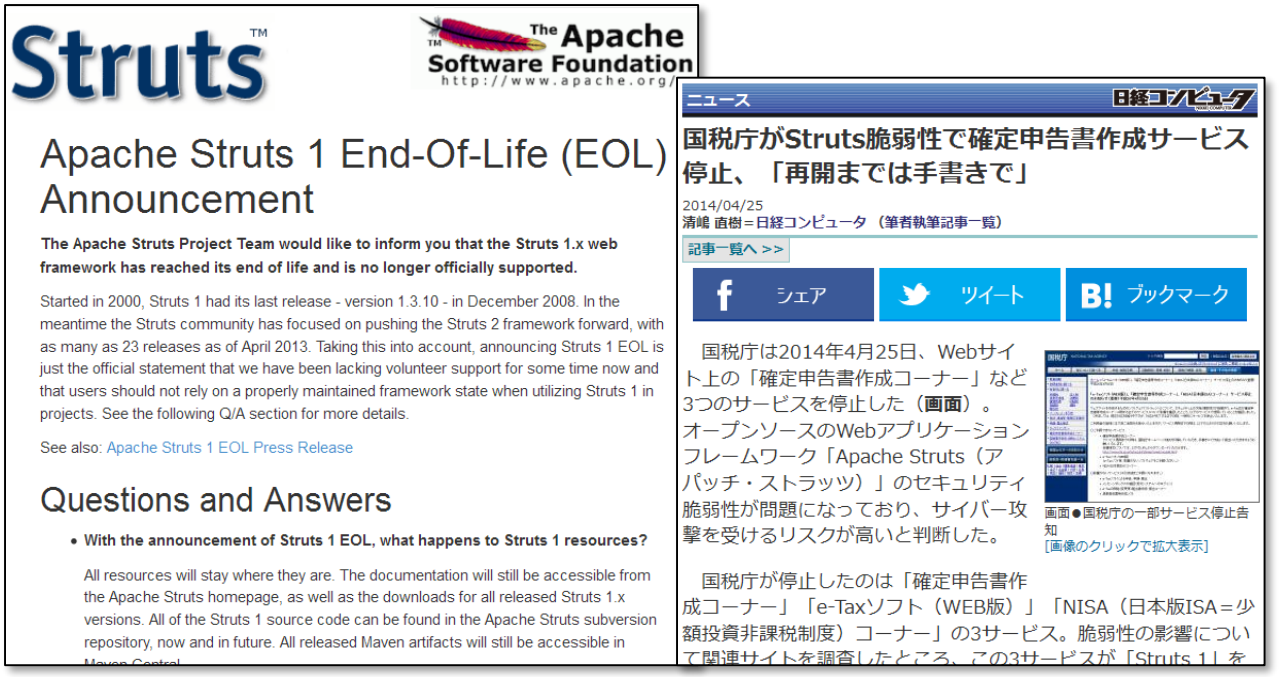
spring source

aspectj crosscutting objects for better modularity

SLF4J Simple Logging Facade for Java

国内で普及率の高かったフレームワークの世代交代

- Struts 1.x
  - 2013年 EoL（End of Life）の発表
  - 2014年 脆弱性発見による混乱（CVE-2014-0114）
- Seaser2
  - 主要開発者のプロジェクト離脱により新機能追加の停止
  - Ajax/HTML5連携やRESTful Webサービス対応の不備



Struts™

The Apache Software Foundation
http://www.apache.org/

Apache Struts 1 End-Of-Life (EOL) Announcement

The Apache Struts Project Team would like to inform you that the Struts 1.x web framework has reached its end of life and is no longer officially supported.

Started in 2000, Struts 1 had its last release - version 1.3.10 - in December 2008. In the meantime the Struts community has focused on pushing the Struts 2 framework forward, with as many as 23 releases as of April 2013. Taking this into account, announcing Struts 1 EOL is just the official statement that we have been lacking volunteer support for some time now and that users should not rely on a properly maintained framework state when utilizing Struts 1 in projects. See the following Q/A section for more details.

See also: Apache Struts 1 EOL Press Release

Questions and Answers

- **With the announcement of Struts 1 EOL, what happens to Struts 1 resources?**

  All resources will stay where they are. The documentation will still be accessible from the Apache Struts homepage, as well as the downloads for all released Struts 1.x versions. All of the Struts 1 source code can be found in the Apache Struts subversion repository, now and in future. All released Maven artifacts will still be accessible in Maven Central.

ニュース 日経コンピュータ

国税庁がStruts脆弱性で確定申告書作成サービス停止、「再開までは手書きで」

2014/04/25
清嶋 直樹＝日経コンピュータ （筆者執筆記事一覧）

記事一覧へ >>

シェア ツイート B! ブックマーク

国税庁は2014年4月25日、Webサイト上の「確定申告書作成コーナー」など3つのサービスを停止した（**画面**）。オープンソースのWebアプリケーションフレームワーク「Apache Struts（アパッチ・ストラッツ）」のセキュリティ脆弱性が問題になっており、サイバー攻撃を受けるリスクが高いと判断した。

画面●国税庁の一部サービス停止告知
[画像のクリックで拡大表示]

国税庁が停止したのは「確定申告書作成コーナー」「e-Taxソフト（WEB版）」「NISA（日本版ISA＝少額投資非課税制度）コーナー」の3サービス。脆弱性の影響について関連サイトを調査したところ、この3サービスが「Struts 1」を

2016年のWebアプリケーション事情

- Java EE提供の各種仕様の機能増強・完成度の向上
  「標準仕様で十分」

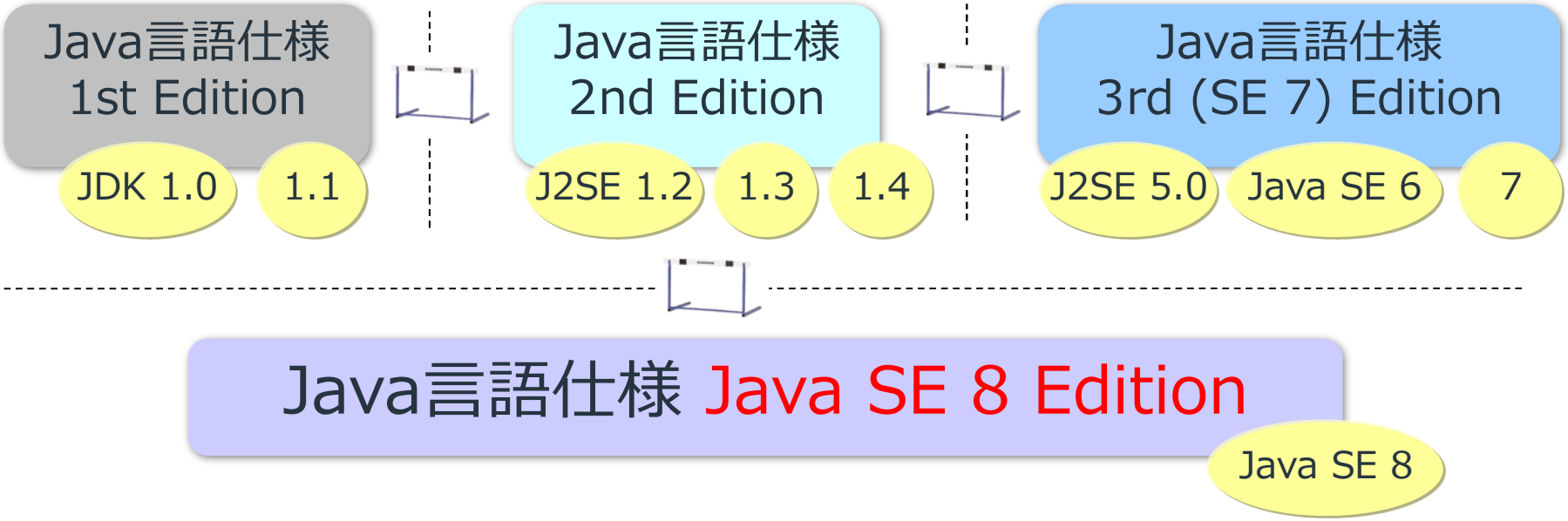
  - CDI 1.x　：依存性注入
    - HibernateやSeaser2の提供していた機能をとりこみ
    - 1.1/1.2で他仕様との親和性が向上
  - JSF 2.x　：Webアプリケーション・フレームワーク
  - JPA 2.x　：O/Rマッピング
    - いずれもCDIとの親和性が向上

→ Java EEへの回帰

Java SE 8

Java Standard Editionの3度目の大変革

- Java SE 8 では、言語仕様レベルの大きな変更が行われた
  - Project Lambda
- 過去2回（1.1→1.2および1.4→5.0）に匹敵する、あるいは、それ以上のインパクトのある大変革



Project Lambda

- Lambda式
- 型推論
- Method Reference
- Default Method
- Stream API

```
( 引数 ) -> { 処理 }
```

オブジェクトとして「変数に代入」したり
「メソッドの引数にわたす」ことができる「コード片」

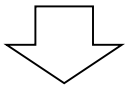
Lambda式とは

- 関数型インターフェースを実装した
  匿名クラスの定義を簡単に行うための糖衣構文

```
Collection<String> texts = ... ;

texts.forEach(new Consumer<String>() {
    @Override
    public void accept(String text) {
        System.out.println(text);
    }
});
```

匿名クラスの定義



```
Collection<String> texts = ... ;

texts.forEach(
    text -> System.out.println(text);
);
```

Lambda式による
簡易記法

Lambda式とStreamによるループの内部化

- 外部イテレータ
  - コンテナから順次、要素を取り出して処理を記述する

```
List<Person> list = ... ;
List<String> ret = new ArrayList<String>();

for (Person p : list) {
        if (p.getGroupId().equals(group)) {
                String name = p.getName();
                ret.add(name);
        }
}
return ret;
```

filter
map
collect

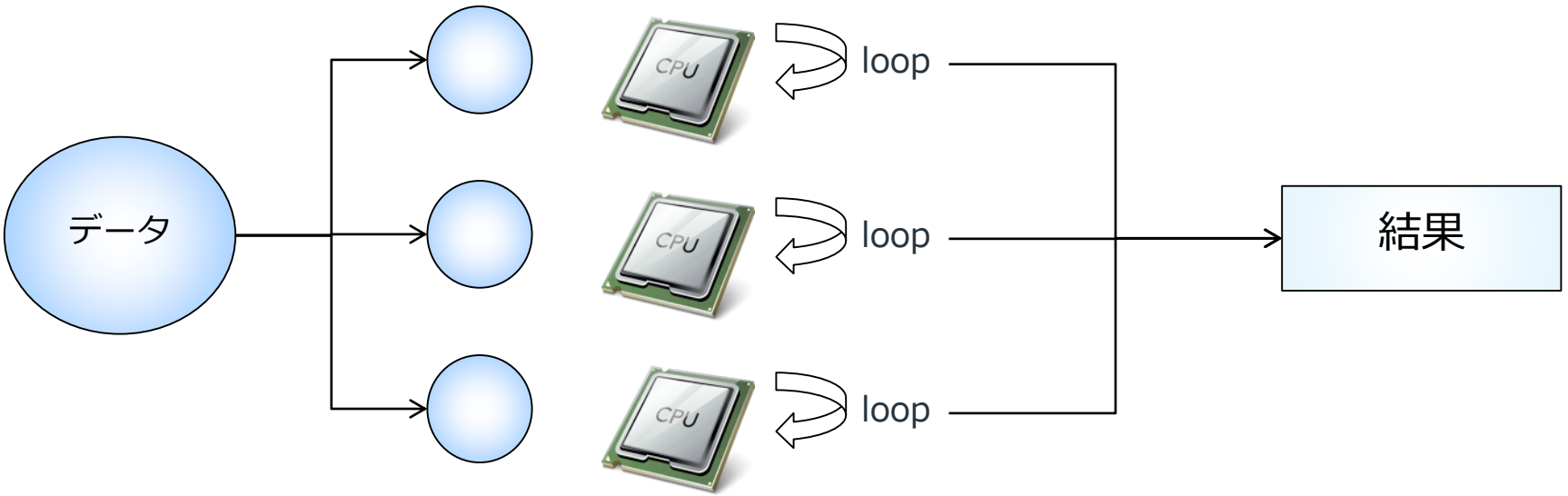
- 内部イテレータ
  - 処理をコンテナにわたして、コンテナが要素ごとに実行する

```
return list.stream()
        .filter(p -> p.getGroupId().equals(group))
        .map(p -> p.getName())
        .collect(Collectors.toList());
```

(参考) Streamによるパラレル処理

- 簡単に並列処理を記述できる

```
// 最もSalaryの高い非Managerをかえす
public static Person getHighestNonManagerEarner(List<Person> list) {
    return list.stream().parallel()
                .filter(p -> !p.isManager())
                .reduce(dummy, (p1, p2) -> (p1.getSalary() > p2.getSalary())? p1 : p2);
}
```



その他 Java SE 8の変更点

- JSR 310 Date & Time：新しい日付API
  - 新しい java.time パッケージ
  - java.util.Date / Calendar APIを刷新
  - 豊富な機能 / クラスもたくさん
- Enhanced Metadata：アノテーションの機能強化
  - 型へのアノテーション / 繰り返しアノテーションをサポート
- Nashorn JavaScript Engine
  - これまでのRhinoよりも、高速で、ECMAScript互換性の高い Nashorn JavaScript Engine

Java SE 6&7との互換性

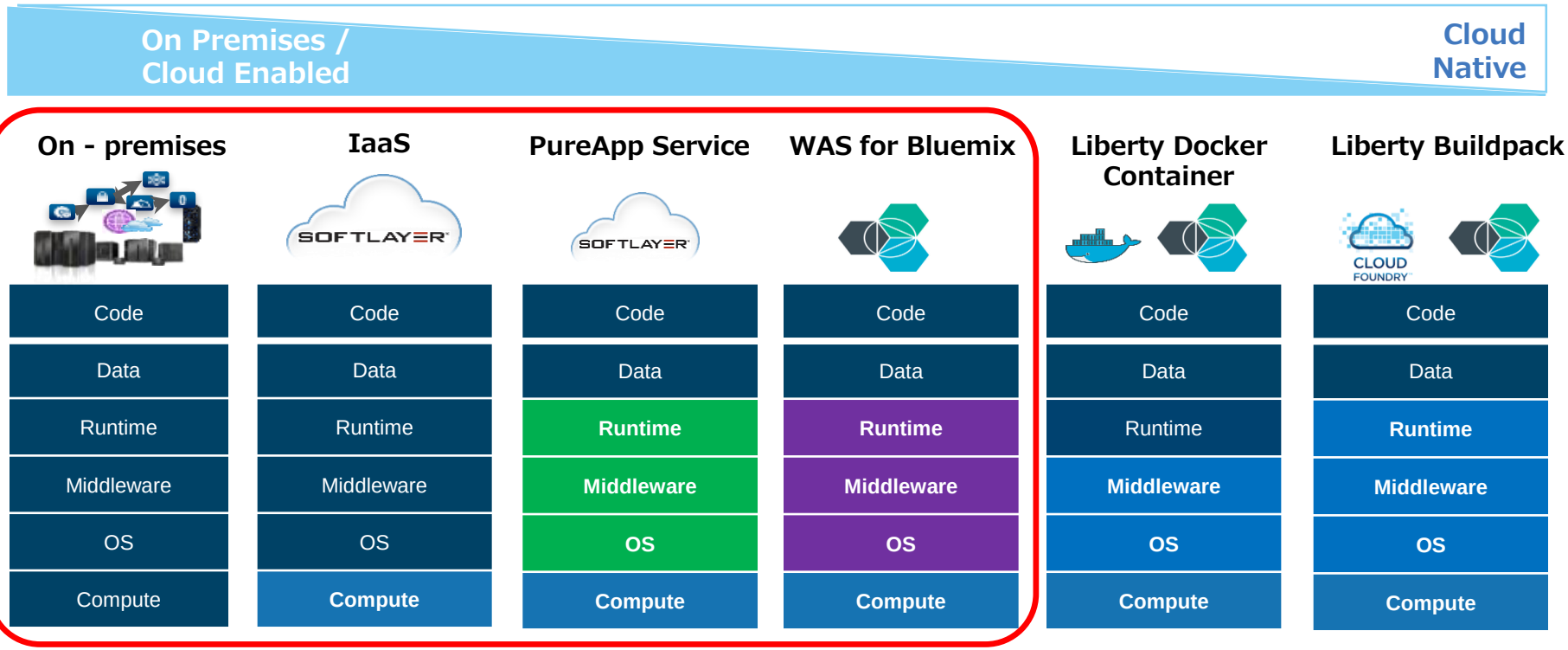
- Java SE 8 と Java SE 7
  - 言語仕様の変革（Lambda式のコーディング、新しい日付API、アノテーション強化）
  - ほぼ全ての既存プログラムが、変更を加えずにJava SE 8で動作する見込み
  - 非互換性の詳細
    - JDK8の互換性ガイド
    - http://www.oracle.com/technetwork/jp/java/javase/overview/8-compatibility-guide-2156366-ja.html
- Java SE 7 と Java SE 6
  - Exceptionをcatch節の中でRe-throwする際の言語仕様変更など軽微なもののみ
  - ほぼ全ての既存プログラムが、変更を加えずにJava SE 7で動作する見込み
  - 非互換性の詳細
    - Java SE 7 and JDK 7 Compatibility
    - http://www.oracle.com/technetwork/java/javase/compatibility-417013.html

tWAS クラウド対応

クラウド対応

Optimize

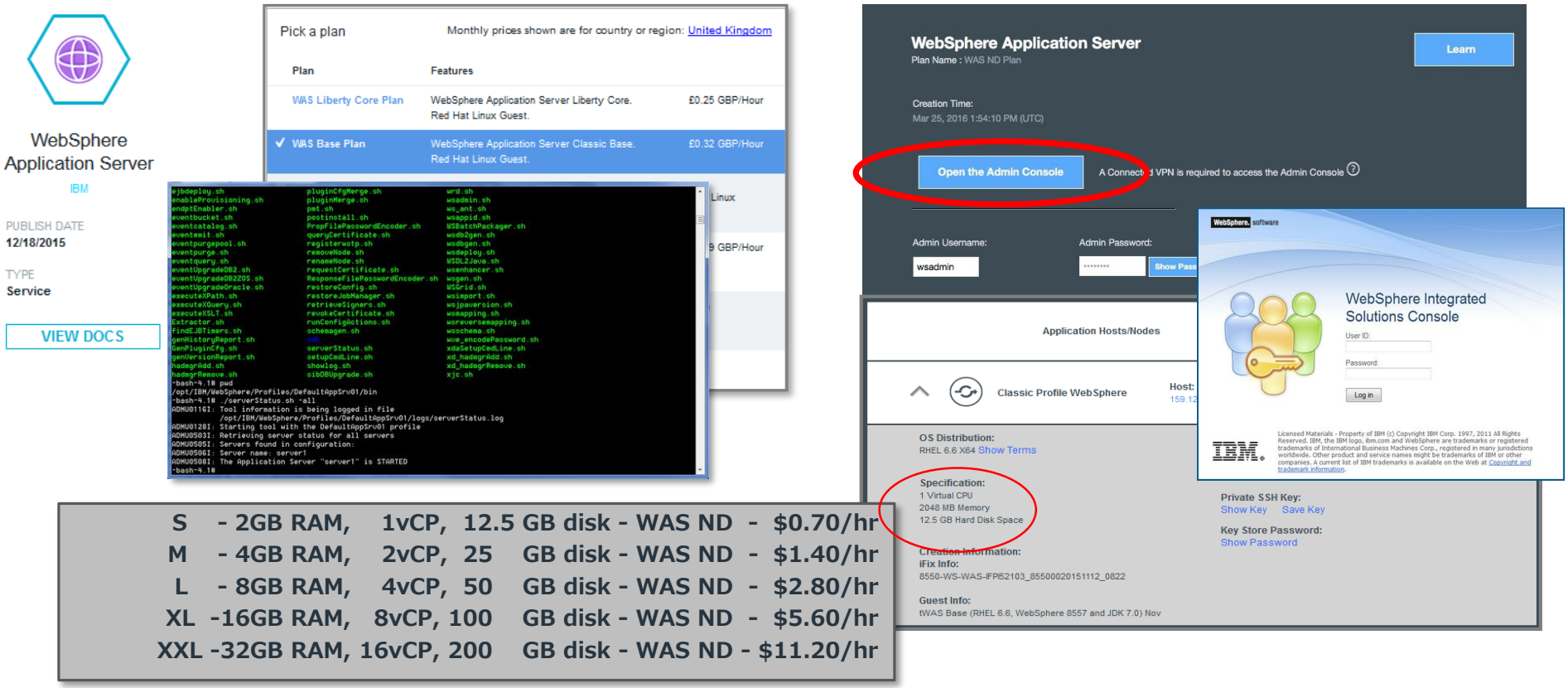
- tWASもさまざまなクラウドで稼動可能
  - IaaS (SoftLayer, AWS, Azure)
  - PaaS (WAS for Bluemix, PureApplication)



WAS for Bluemix

Optimize

- WAS for Bluemixでも同時にWAS V9を提供予定
  - tWAS V9ベータは、WAS for Bluemix上でも提供



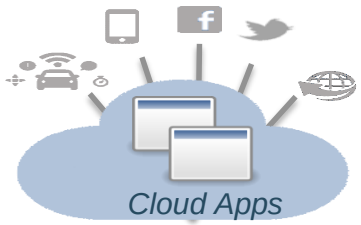
S	- 2GB RAM,	1vCP,	12.5 GB disk	- WAS ND	- $0.70/hr
M	- 4GB RAM,	2vCP,	25 GB disk	- WAS ND	- $1.40/hr
L	- 8GB RAM,	4vCP,	50 GB disk	- WAS ND	- $2.80/hr
XL	-16GB RAM,	8vCP,	100 GB disk	- WAS ND	- $5.60/hr
XXL	-32GB RAM,	16vCP,	200 GB disk	- WAS ND	- $11.20/hr

Variable VM sizes

WAS for Bluemixシングル・テナント

Optimize

シングル・テナント:
- Dedicated Server – セキュリティ要件に合致
- 高パフォーマンス
- Bluemixとお客様データセンターの直接接続 (up to 10 GB)
- Bluemix 管理コンソール
- Bluemix Services



マルチ・テナント:
- 費用対効果 (Shared Server – Multiple/Secure VMs)
- Bluemix 管理コンソール
- Bluemix Services

tWAS Dockerイメージの提供

- 6/15 tWASとIHSのDockerイメージを提供

WebSphere traditional and IBM HTTP Server on Docker Hub

KAVISURESH / JUNE 15, 2016 / 2 COMMENTS

We are pleased to announce that pre-built Docker images containing IBM WebSphere Application Server for Developers V8.5.5.9 traditional and IBM HTTP Server are now available on Docker Hub.

This enables you to get WebSphere Application Server traditional and IBM HTTP Server up and running quickly in your Docker environment with only a single command.

Running WebSphere Application server traditional is as simple as executing:

```
docker run --name test -h test -p 9043:9043 -p 9443:9443 -d ibmcom/websphere-traditional
```

RECENT POSTS

WebSphere traditional and IBM HTTP Server on Docker Hub

Beta: WebSphere Liberty and tools (June 2016)

Putting the 'Micro' into Microservices with Raspberry Pis

Running a multi-container application using Docker Compose

Deploying applications to WAS traditional and IBM HTTP Server under Docker

WASdev「WebSphere traditional and IBM HTTP Server on Docker Hub」
https://developer.ibm.com/wasdev/blog/2016/06/15/websphere-traditional-ibm-http-server-docker-hub/

V8.0からのアップデートとトポロジー

V8.5.xでの重要なアップデート

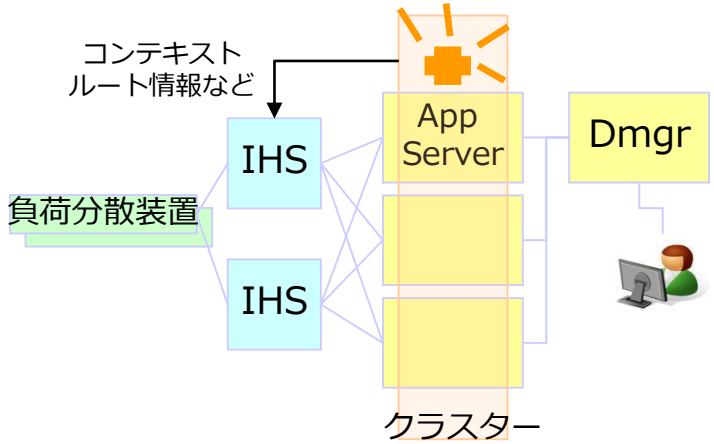
- V8.5
  - インテリジェント管理
    - 旧Virtual EnterpriseのNDへの統合
- V8.5.5
  - WebサーバーPlugin ODR
    - ODRで提供される機能のほとんどがIHSのプラグインで実現可能に
  - インメモリー・キー・バリュー・データストア(KVS)
    - WXS (WebSphere eXtreme Scale) の同梱
      - ND版は、フル機能利用可能
      - Base版は、HTTPセッションと動的キャッシュの保管のみ可能

V8.0とV9.0を比較すると、さまざまな機能追加

インテリジェント管理 (ND版のみ)

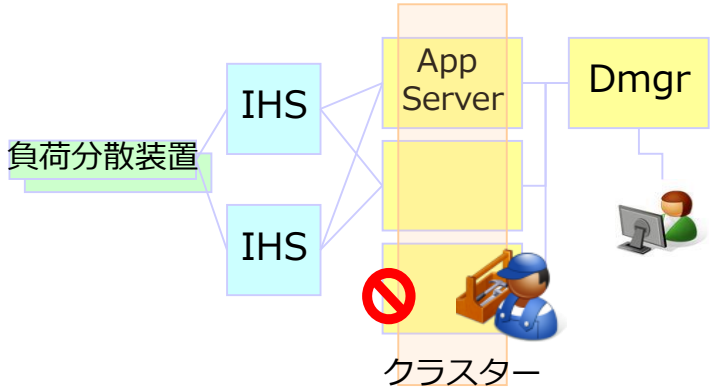
❑ 煩雑な運用管理を自動化

①アプリケーションの導入削除の自動検知



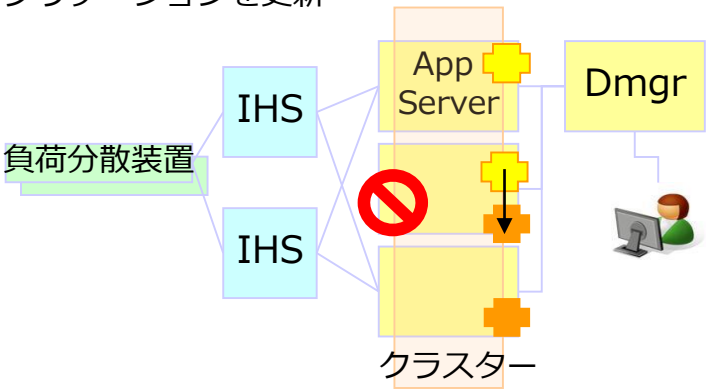
②保守モード

メンテナンスのためのサーバー切り離し



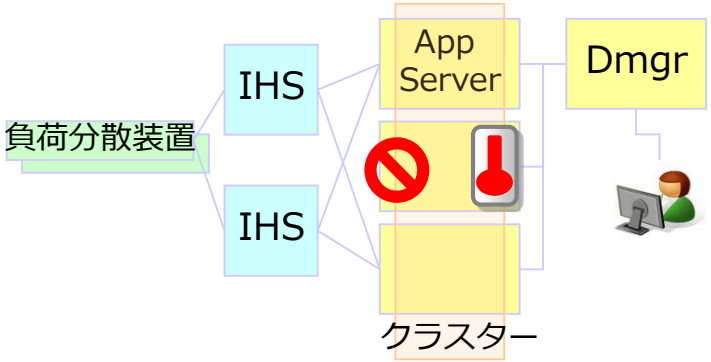
③アプリケーションの無停止ロールアウト

Webサーバーで要求フローを制御しながら、
アプリケーションを更新



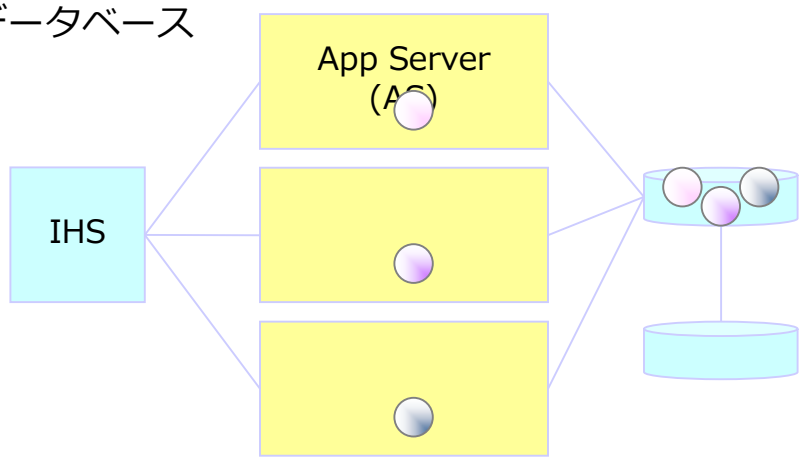
④ヘルス・ポリシー

障害兆候を事前に検知し自動切離し、問題判別資料を取得
（メモリー・リーク傾向、応答時間超過など）

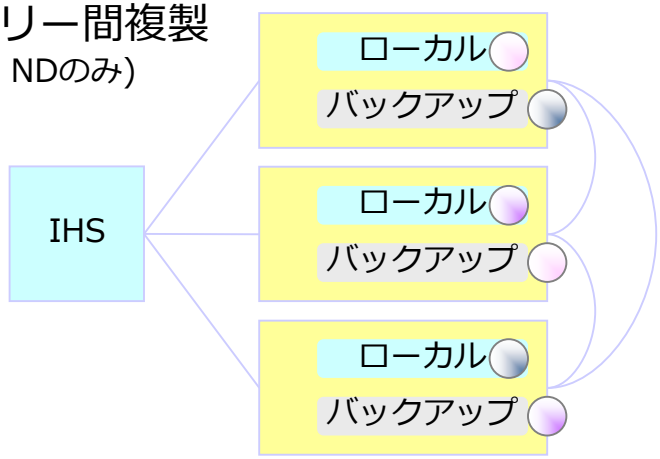


WXSによるHTTPセッションの複製

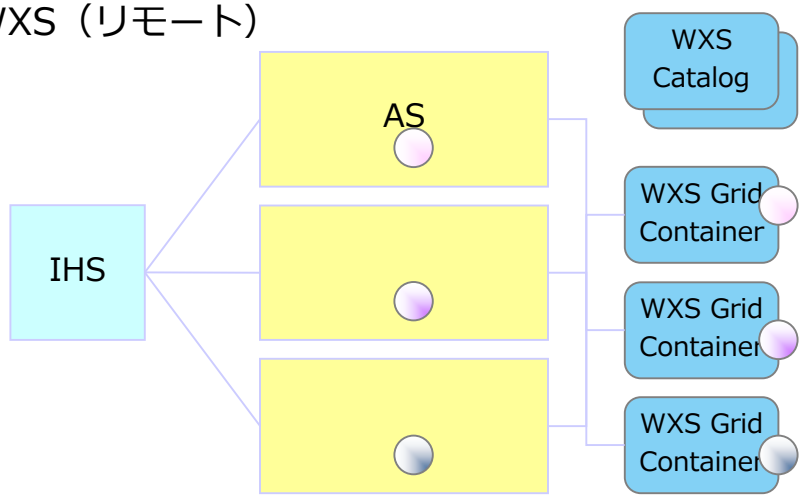
- データベース



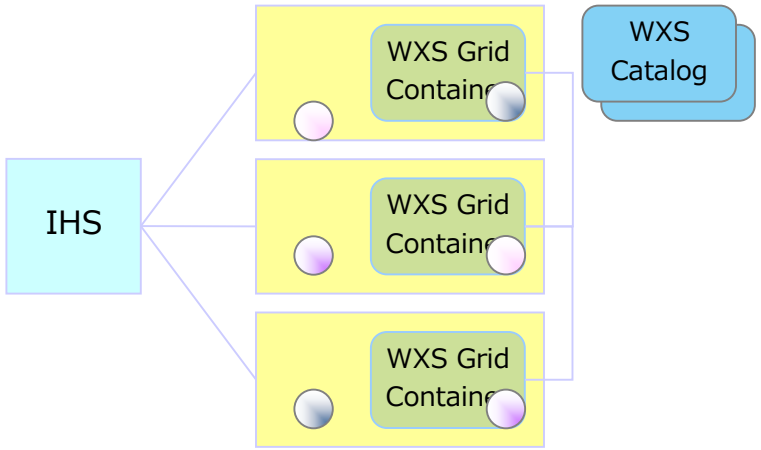
- メモリー間複製
(WAS NDのみ)



- WXS（リモート）



- WXS（組込み）

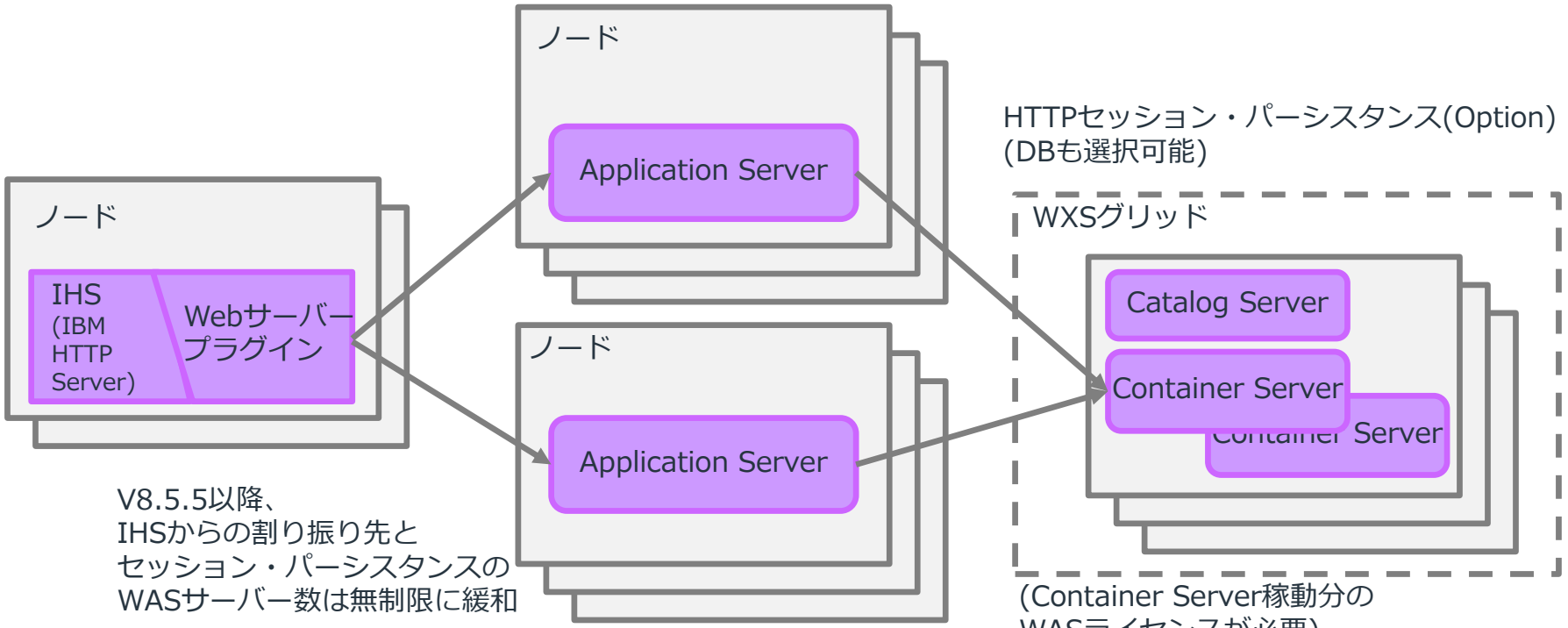


(参考) HTTPセッションの複製方式比較

セッション保管先	データベース	メモリー間複製 (Mem2Mem)	WXS連携	
			リモート構成	組込み構成
WAS Base サポート	○	× WAS ND構成のみ	○	
アプリケーション への修正	– なし	- なし	△ コード修正は不要。Webモジュールの META-INFに構成ファイルのみ配置。	
構築	△ WASに加え、別途DBの構築・構成、高可用性セットアップが必要	◎ WASコンソールからの設定	△ WASに加え、WXS関連プロセスを別途構成	○ WASに加え、別途カタログ・サーバーを別途構成
管理	△ DBの管理が別途必要	◎ WASのみ	△ 管理対象プロセスが増える	○ カタログサーバーを追加で管理
アプリケーションとの Java ヒープ共有	○ (なし)	× (共有)	○ (なし)	× (共有)
セッション・サイズ	△ セッション内のエントリー最大2MB	○ サイズ制限なし (ヒープ・サイズによる制約のみ)	◎ サイズ制限なし（ヒープ・サイズによる制約のみ、プロセス数を増やすことで対応）	
信頼性	○ Best Effort	○ Best Effort	◎ 同期モード（書込保障）、非同期モードの選択が可能	
複製の数	○ DB	○ 1つ（単一レプリカ）またはクラスター全体	○ 複製数（同期、非同期）を指定可能	
保管先障害時の 複製データの復旧	–	△ 障害後、更新があったエントリーのみ復活	○ WXSにより自動復旧	
拡張性	△ Scale Upアプローチ	△ クラスター・メンバーの追加	○ WXSプロセスの追加により、リニアにスケール	
データ･センター間複製	×	×	○ ゾーン設定により可能	

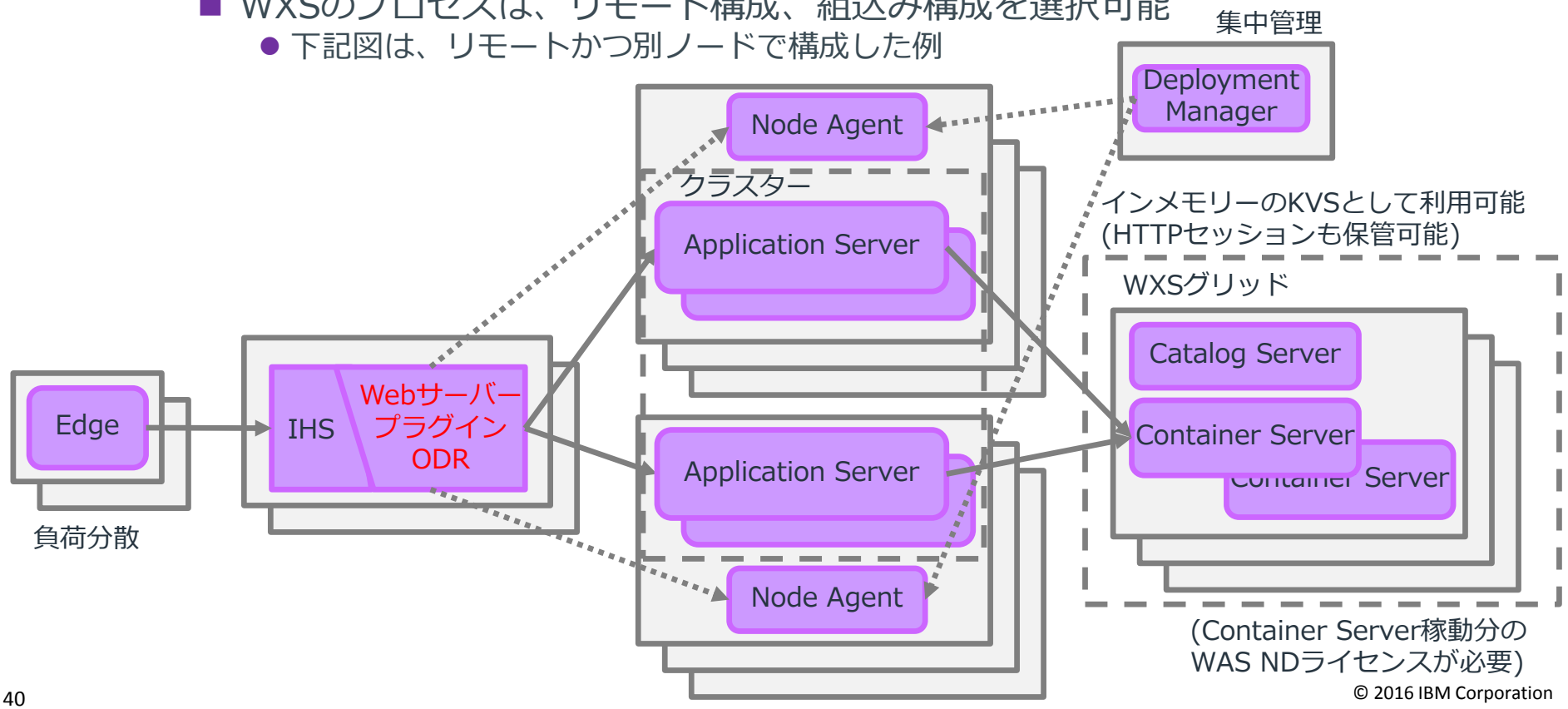
Base版のトポロジー構成例

- スタンドアロン or シンプル・クラスター
  - V8.xやV7と大きな違いなし
  - HTTPセッションの格納先として、WXSを利用可能
    - WXSのプロセスは、リモート構成、組込み構成を選択可能
      - 下記図は、リモートかつ別ノードで構成した例



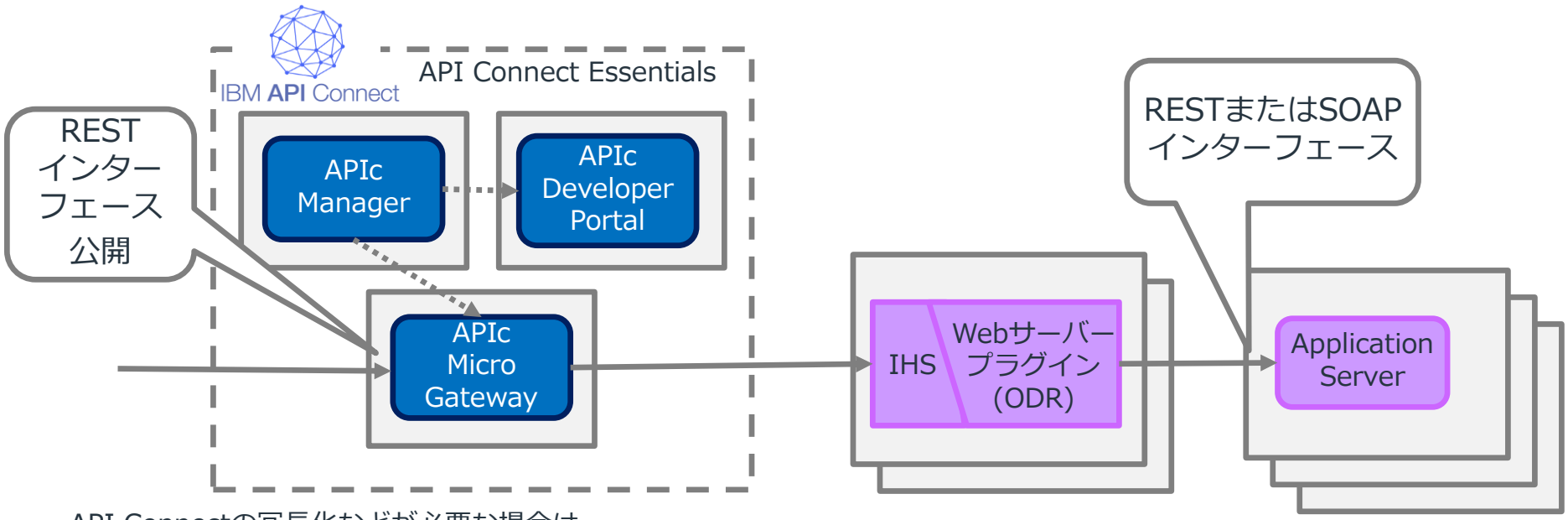
ND版のトポロジー構成例

- NDクラスター
  - V8.0までと比較すると、インテリジェント管理機能が使用可能
    - それもV8.0までとトポロジーの変更はなく、IHSでODRの機能を稼動させることが可能
  - キャッシュやKVSとして、WXSを活用可能
    - WXSのプロセスは、リモート構成、組込み構成を選択可能
      - 下記図は、リモートかつ別ノードで構成した例



API Connectを組み合わせた構成

- tWASでもAPI Connectと連携
  - WASが公開するAPIをAPI Connectを利用して、APIのライフサイクルを一元管理/制御
    - ポリシーの適用(流量制御など)
    - アクセス制御



API Connectの冗長化などが必要な場合は、
上位エディションのAPI Connect Professional
またはEnterpriseエディションが必要

(APIcコンポーネント稼動分の
WASライセンスが必要)

まとめと参考資料

まとめ

- WAS V9 traditional アップデート
  - Java EE 7 完全対応
    - これまで同様の、運用管理、可用性、拡張性とともに、最新のプログラミング・モデルを利用可能に
  - Java SE 8 対応
  - API Connect 同梱 (WebSphere Connect)
  - クラウド対応
    - WAS for Bluemix：tWASもBluemix上で提供
      - V9リリース、シングル・テナント

参考資料

- Knowledge Center “WebSphere Application Server Network Deployment traditional V9”
  - http://www.ibm.com/support/knowledgecenter/en/SSAW57_9.0.0/as_ditamaps/was900_welcome_ndmp.html
- WASdev ”WebSphere Liberty and tools 16.0.0.2 release”
  - https://developer.ibm.com/wasdev/blog/2016/06/24/websphere-liberty-and-tools-16-0-0-2-release/
- Oracle “Java EE 7 のテクノロジー”
  - http://www.oracle.com/technetwork/jp/java/javaee/tech/index.html
- developerWorks-Japan “WebSphere Application Server (WAS)”ポータル
  - http://www.ibm.com/developerworks/jp/websphere/category/was/
- developerWorks-Japan ”Java EE 7 アプリケーション設計ガイド - WebSocket編”
  - http://www.ibm.com/developerworks/jp/websphere/library/was/javaee7_appguide/1.html
- developerWorks-Japan ”Java EE 7 アプリケーション設計ガイド - JAX-RS 2.0 編”
  - http://www.ibm.com/developerworks/jp/websphere/library/was/javaee7_appguide/5.html
- developerWorks-Japan “Java 8 言語での変更内容”
  - http://www.ibm.com/developerworks/jp/java/library/j-java8lambdas/
- developerWorks-Japan “JVM の並行性: Java 8 での並行処理の基礎”
  - http://www.ibm.com/developerworks/jp/java/library/j-jvmc2/

WebSphere Application Server