

このセッションでは、WASの最新の強化ポイントや、Libertyの特長について説明します。

IBM

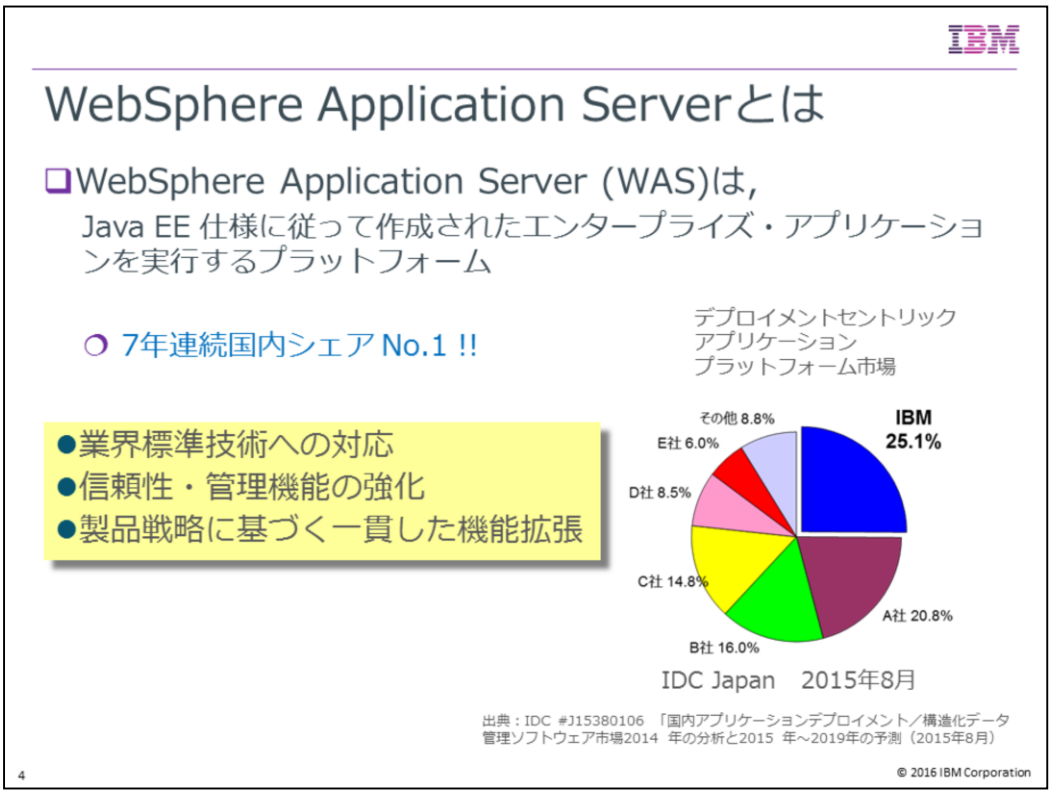
アジェンダ

- WASの最新動向 & Liberty とは
- Liberty の特長

© 2016 IBM Corporation



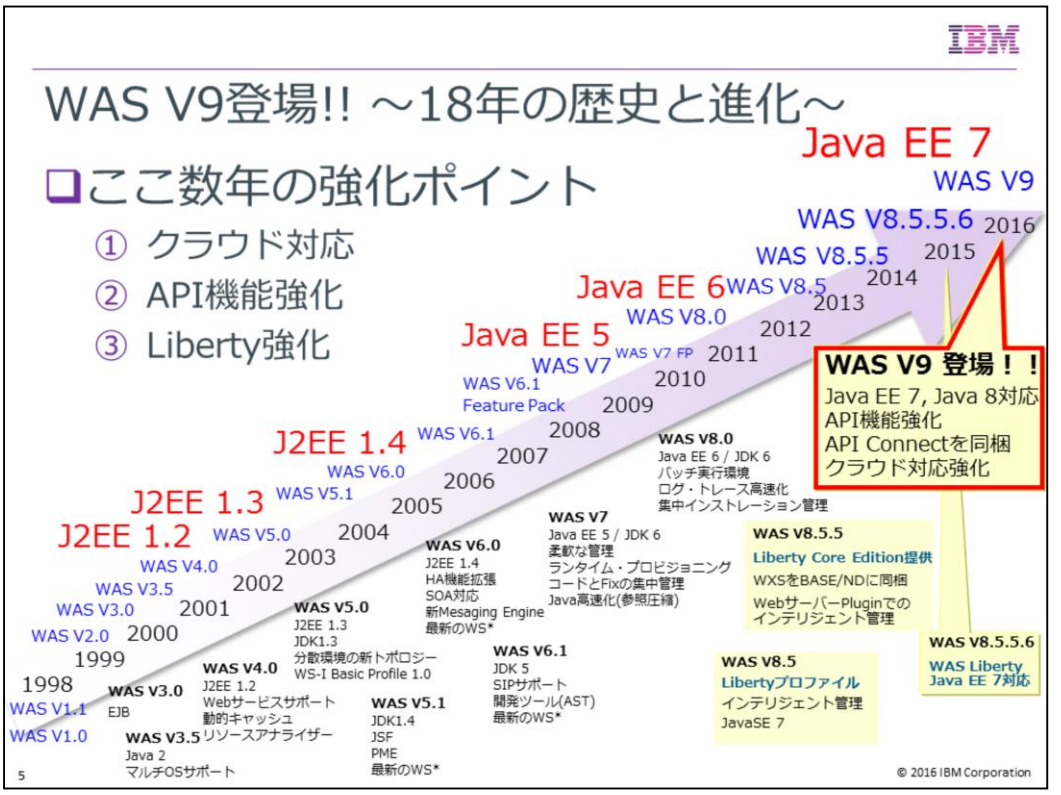
この章では、WASの最新動向として、ここ数年強化されているポイントについてご紹介します。



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です。

WASは、製品戦略に基づき一貫して自社で開発を継続しています。Java EE に代表される業界標準技術への対応、信頼性・管理機能の強化を主に行ってきています。

市場シェアも高く、2015年8月に発表されたIDCの調査結果によると2014年の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で、シェアNo.1を獲得しています。IDCの調査結果は、2014年だけでなく、2008年から7年連続で国内シェアNo.1の獲得しています。

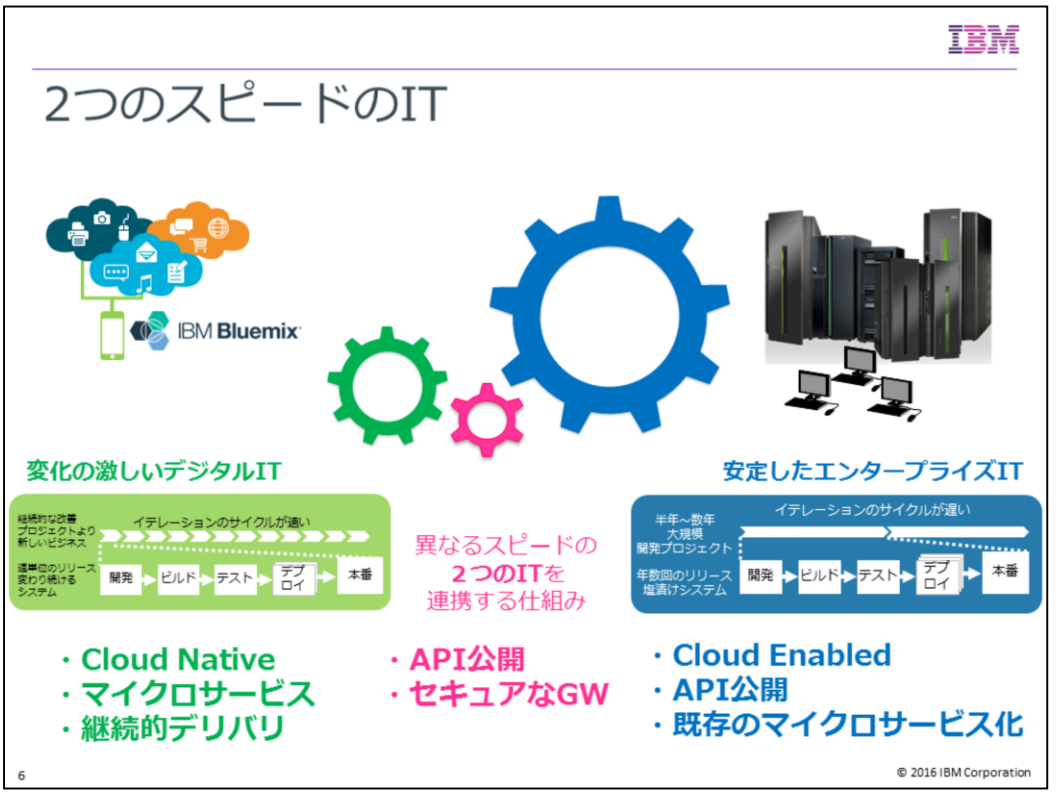


WASは1998年にV1がリリースされ、今年で18年の歴史がある製品です。

今年の6月には、最新版のV9がリリースされています。ここ数年WASは、Java SE、Java EEに代表される標準仕様対応、信頼性・管理機能強化に加えて、最近のシステムの動向を踏まえた機能強化が図られています。

特に、①クラウド対応、②API機能強化、③新しいランタイムLiberty強化、が主な強化ポイントです。

Libertyランタイムは、2012年にリリースされたWAS8.5から、これまでのWASのランタイムに加えて提供されています。これまでのWASは、traditional ランタイムや tWAS と言われています。



①クラウド対応、②API機能強化、③新しいランタイムLiberty強化、が図られた背景を簡単に説明します。現在、企業のITは、2スピードIT(Bimodal IT)と呼ばれるように、二極化しています。

図左のようなエンド・ユーザーに近いシステム「変化の激しいデジタルIT」では、ユーザーの要望に即座に対応することが求められるため、素早い開発、継続的デリバリーが必要になっています。こうしたニーズから、以下のようなアプローチが取られます。

開発手法：継続的デリバリーを実現できる、アジャイル開発や、DevOpsのアプローチが主に使用されています。

アプリケーション：スピード重視のため、機能毎にアプリケーションを実装し、複数のサービスを組み合わせる、マイクロサービス・アーキテクチャーのアプローチが増えてきています。単機能毎のアプリケーションは、機能変更に強く、テストも最小単位で実施が可能で、SoEのスピード重視にマッチします。

インフラ：すぐに利用可能なクラウド環境が活用されています。インフラ構築を簡単に行う工夫として、インフラ構築をコード化するInfrastructure as codeや、一度作成した環境には手を入れず、常に新しい環境を作成する、Immutable Infrastructureの考え方を採用されるケースも増えています。

一方、図右のような基幹システムに代表される企業内のシステム「安定したエンタープライズIT」は、以下のようなアプローチが取られます。

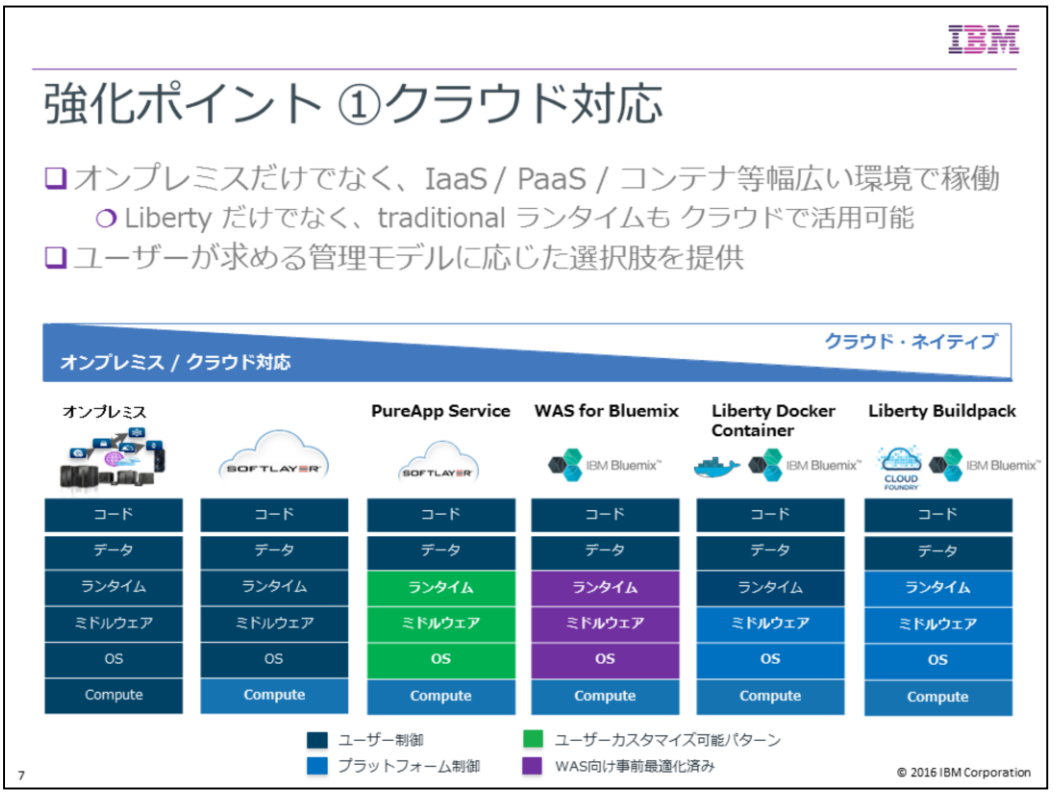
開発手法：多くの場合ウォーターフォールでの開発がされています。

アプリケーション：要件変更も少ないため、巨大なアプリとなりやすいモノリシック・アプリケーションで実装されています。

インフラ：オンプレミスへのデプロイが行われています。

しかし、基幹システムが保持している情報をエンド・ユーザーに素早く提供したいというニーズから、API公開する方法、マイクロサービス化といった検討も始まりつつあります。

「変化の激しいデジタルIT」と「安定したエンタープライズIT」の間では、2つを組み合わせる仕組みとしてAPIが注目されています。こういった背景から、WASは、①クラウド強化、②API強化、③軽量・高速なランタイムLibertyを提供、強化しています。次ページ以降で、3つの強化ポイントについてご紹介していきます。



WASのクラウド対応としては、オンプレミスだけでなく、IaaS / コンテナ / PaaS等幅広い環境で利用することができます。

図の一番左側はユーザーの自由度が高い反面、管理の手間がかかり、右に行くに従い、ユーザーが制御できる範囲が狭まりますが、その反面、管理の手間が少なくなります。

WASはユーザーが求める管理モデルに応じた様々な選択肢を提供しています。

tWASは、左から4つ目までに対応しています。Dockerに関しては、サポート対象ですが、BluemixのDocker Containerなどで提供はしていません。Libertyに関しては、全ての環境で稼動できます。



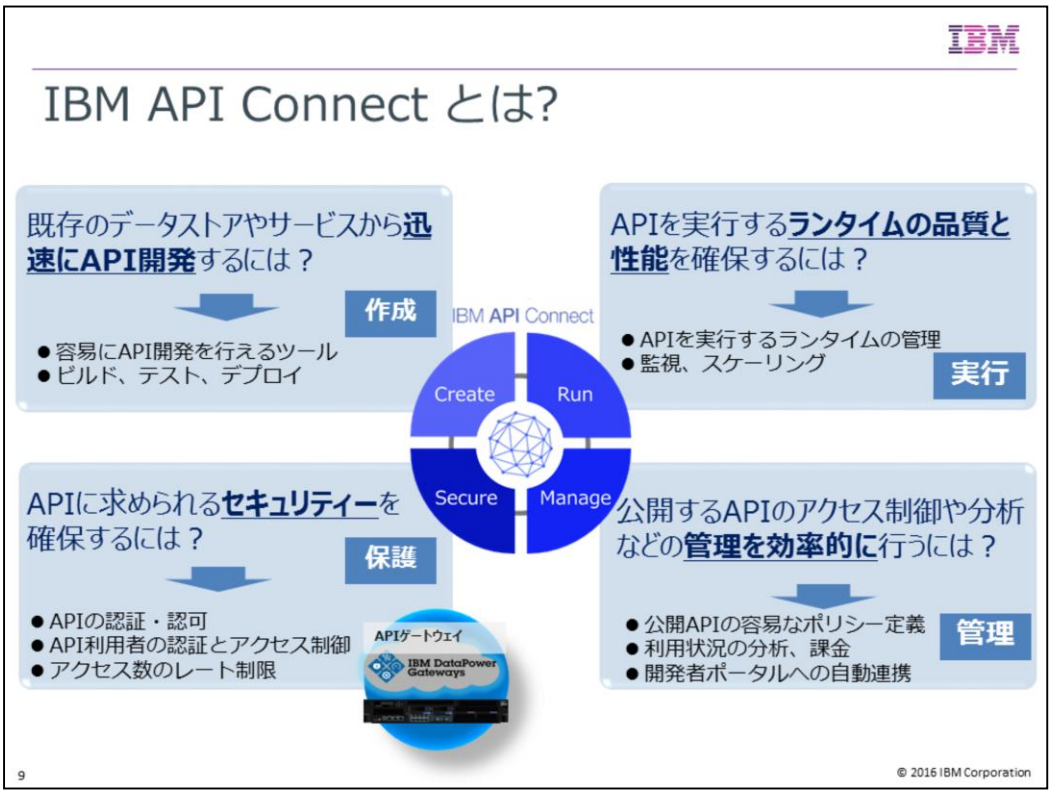
API機能対応としては、特にLibertyでは、API discoveryというAPIを作成・公開する機能、およびBluemixのサービスを利用するための機能が追加されています。

更にtWAS、Libertyかかわらず、WAS8.5.5.9以降のWASに、IBM WebSphere Connect、実体はAPI Connect のEssentials エディションが、同梱されるようになりました。

API Connect Essentialsはサポートがないエディションですが、WASに同梱されるものは、IBMサポートが付き、WASのPVU毎に月額500APIコールできるようになります。

既存JavaアセットをAPI化して、WebSphere Connectにより、APIのライフサイクル管理が実現できます。

WebSphere Connectについて詳しくは次ページ以降で説明します。



WebSphere Connectの実体は、IBM API Connect Essentialsです。

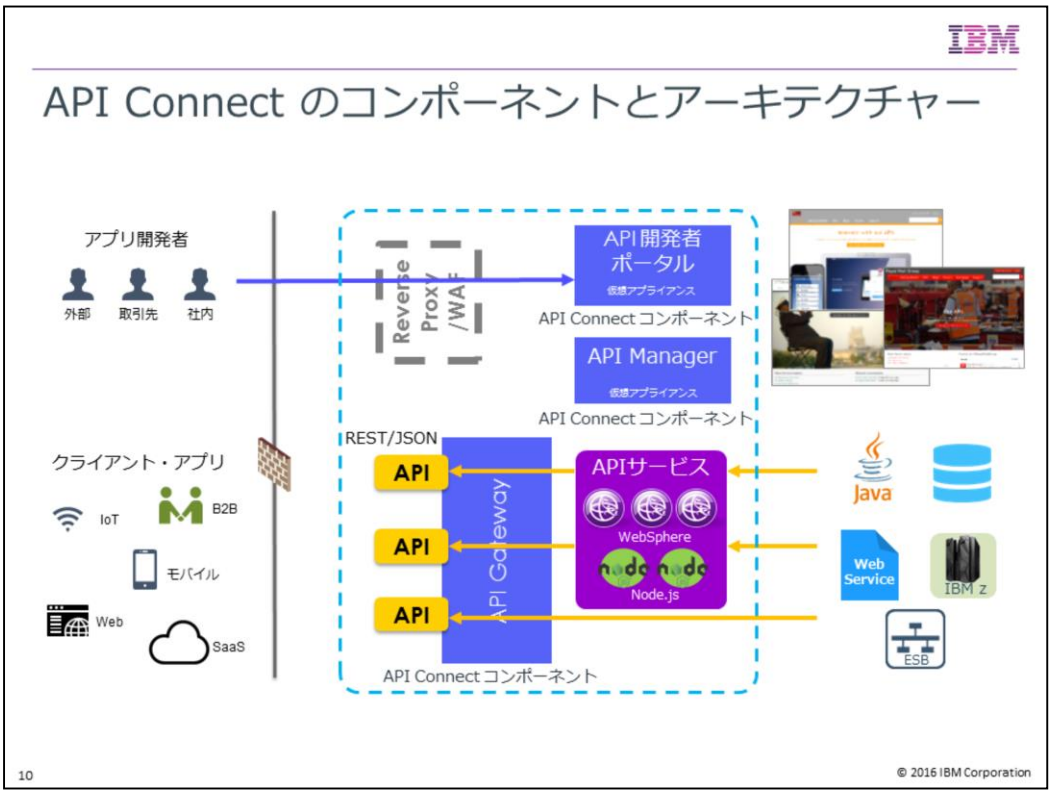
IBM API Connectは、APIの作成、実行、管理、保護を一元的にサポートする製品です。

APIの作成では、WASでもAPIの機能が強化されています。また、昨年買収したStrongLoopから既存のデータストアやサービスから素早くAPIを作成できるDeveloper Toolkitを提供しています。

APIの実行では、実際のAPIを実行するランタイム(WAS、Node)の、管理、監視ができる機能を提供しています。

APIの管理では、公開するAPIの、アクセス制御などのポリシー定義、利用状況の分析、開発者への公開など、API管理に必要となる機能を提供しています。

APIの保護では、APIの求められるセキュリティを確保するため、API ゲートウェイ機能を提供しています。ゲートウェイで、APIの認証・認可、アクセス制御、レート制限などを実現できます。

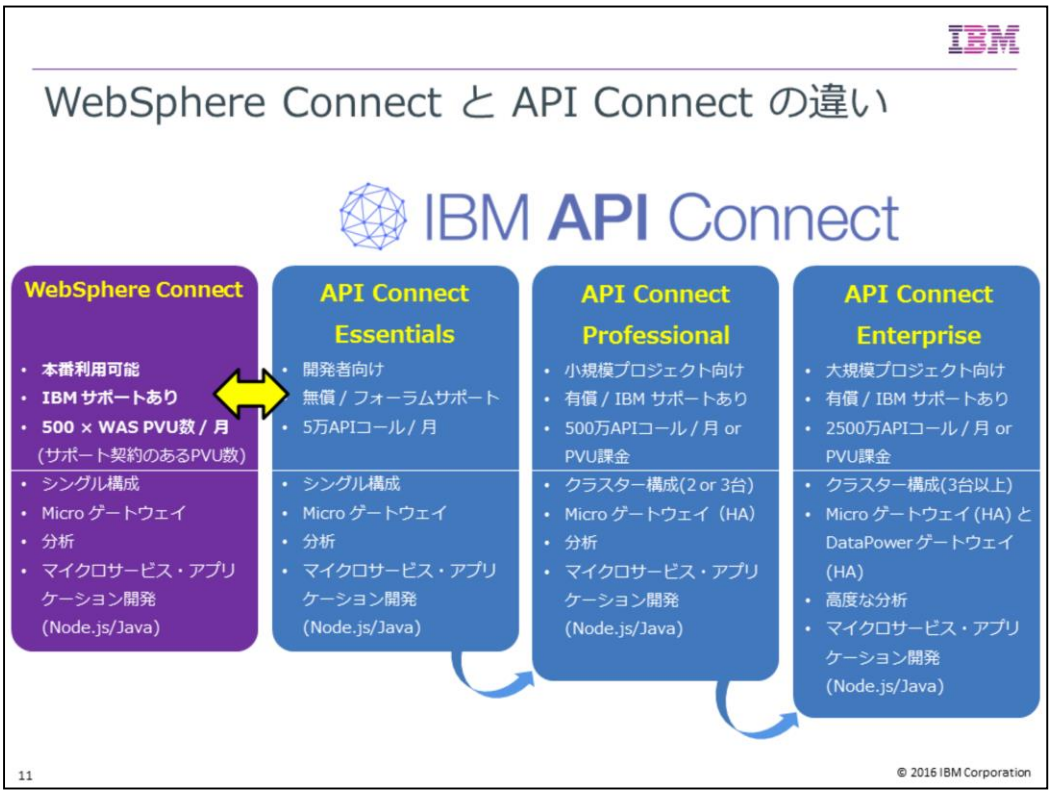


こちらは、API Connectのコンポーネントと、アーキテクチャーを表しています。

APIの実体は、紫の箱で表しており、WASでご用意いただくか、Node.jsで実装することも可能です。

青の箱がAPI Connectのコンポーネントになります。APIへのアクセスを保護するコンポーネントであるAPI Gateway、APIの管理を行うコンポーネントであるAPI Manager、提供したAPIを開発者のために公開するAPI 開発者ポータルというコンポーネントから構成されています。

WASをお持ちのお客様は、こうしたコンポーネント、構成で、APIを公開するための基盤を実現できます。



WebSphere ConnectとAPI Connectの違いを図示しています。

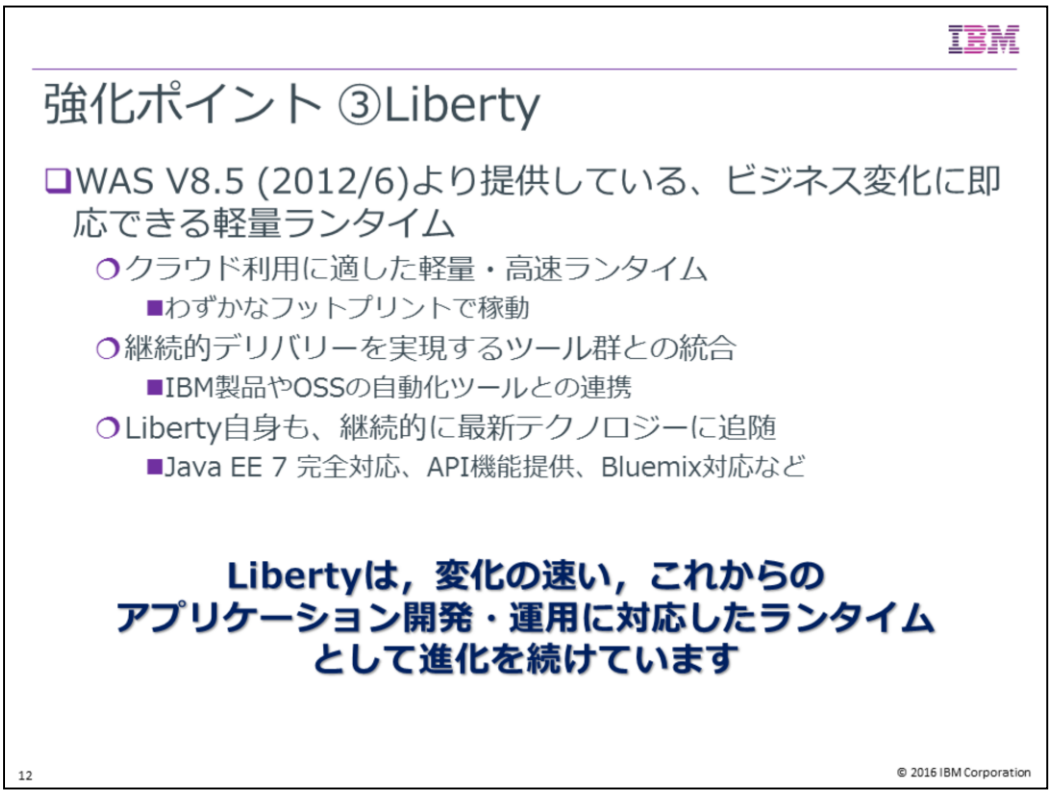
実体は、API Connect Essentialsのため、機能制約は、Essentialsと同じです。ライセンス的に、WebSphere Connectでは、以下の点が補強されています。

・本番利用可能

・IBM サポートあり

・APIコール数が、WASのPVU数に応じて増やすことが可能　500 × WAS PVU数 / 月

WebSphere Connectは 、シングル構成しかとれないため、クラスター構成を行いたいという場合は、Professional以上のライセンスが必要になります。



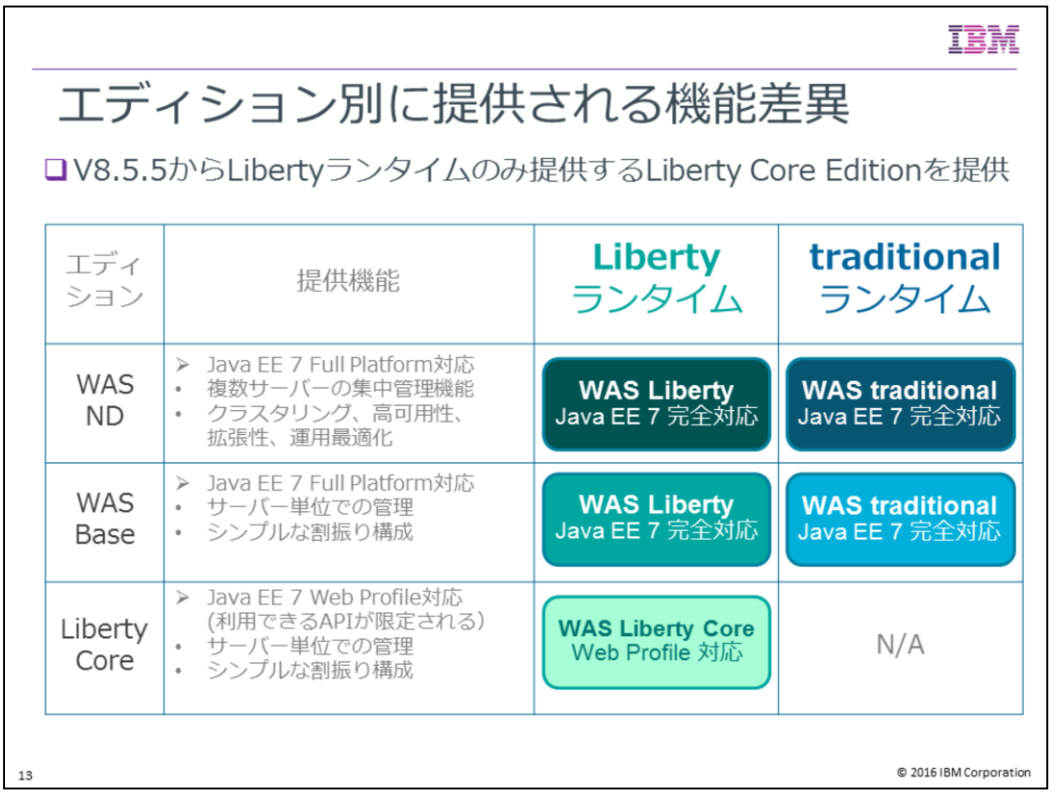
最後に、本セミナーの主題であるLibertyに関してです。

WASはV8.5から、これまでのWASのランタイムのほかに、Libertyという軽量なランタイムを提供しています。

tWASと比較して、軽量で高速起動できる点は、メモリ使用量等のリソース使用量で課金されるクラウド利用に適しています。また、IBM製品、OSS製品の自動化ツールとの統合が行われており、ビジネス・ニーズに即応できる継続的デリバリーを実現できます。

Liberty自身も継続的に最新テクノロジー、新機能が現在も継続的にリリースされています。

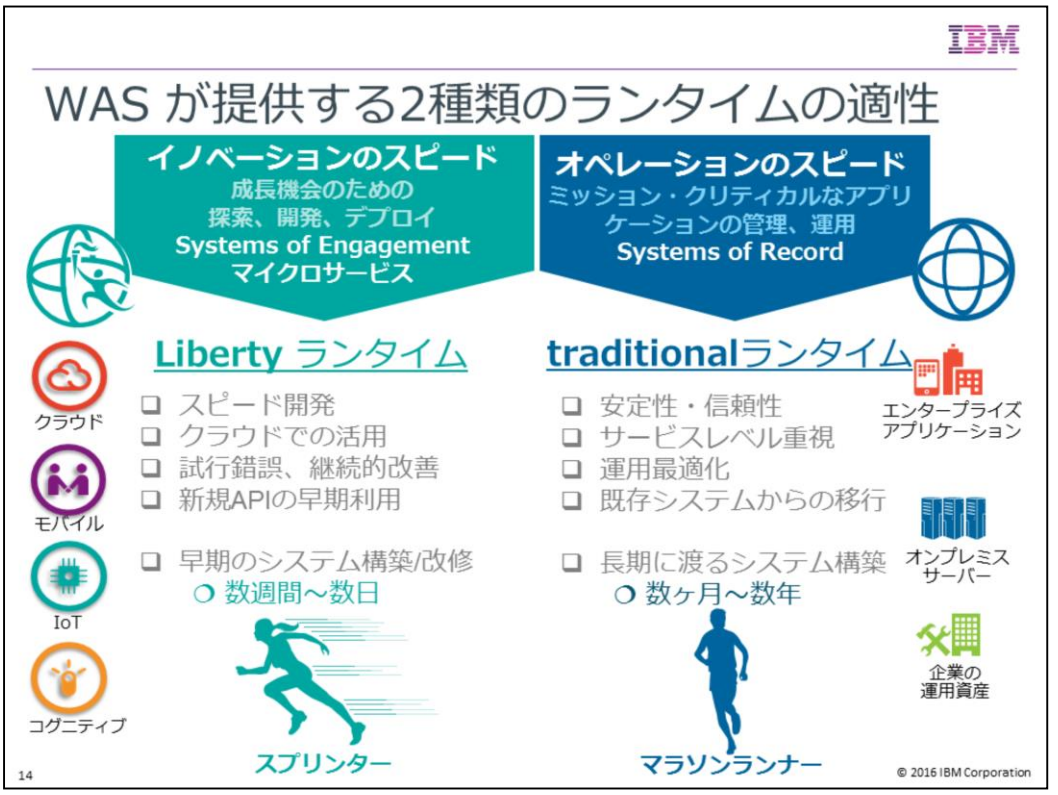
Libertyは、昨今のシステム動向に対応した、変化の速い、アプリケーション開発・運用に対応するランタイムとして、リリースから4年、強化され続けています。



こちらは、Liberty ランタイムと traditional ランタイムのエディション毎に提供される機能の差異を表しています。

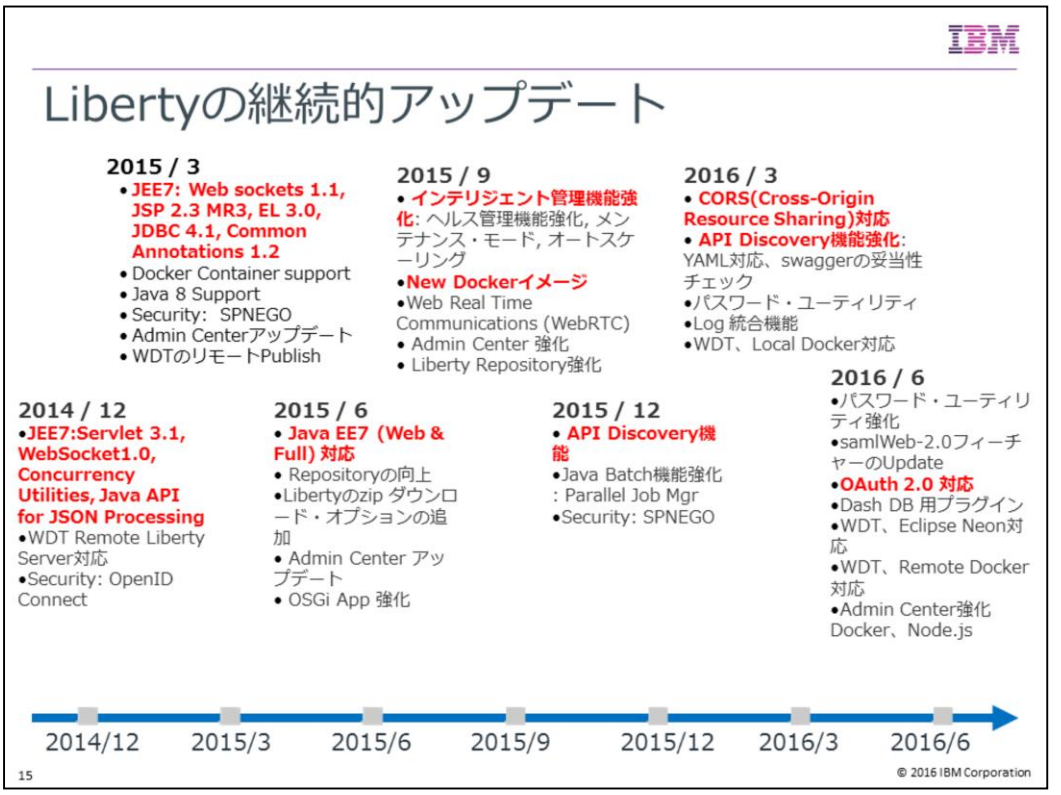
V8.5.5からは Liberty Core EditionというLibertyのみを提供するエディションが提供されています。LibertyはWASの全てのエディションで提供されていますが、エディションによって機能が異なります。Liberty Core エディションは、Java EE 7 の Web Profile 相当の API をサポートしています。Base と ND エディションでは、Libertyランタイムと tWAS ランタイムを提供しており、いずれも Java EE 7 の full Platform の API をサポートしています。

NDエディションでは、Libertyも集中管理機能、クラスタリングによる高可用性を実現できます。



こちらは、Libertyランタイムとtraditionalランタイムの主な利用用途を表しています。

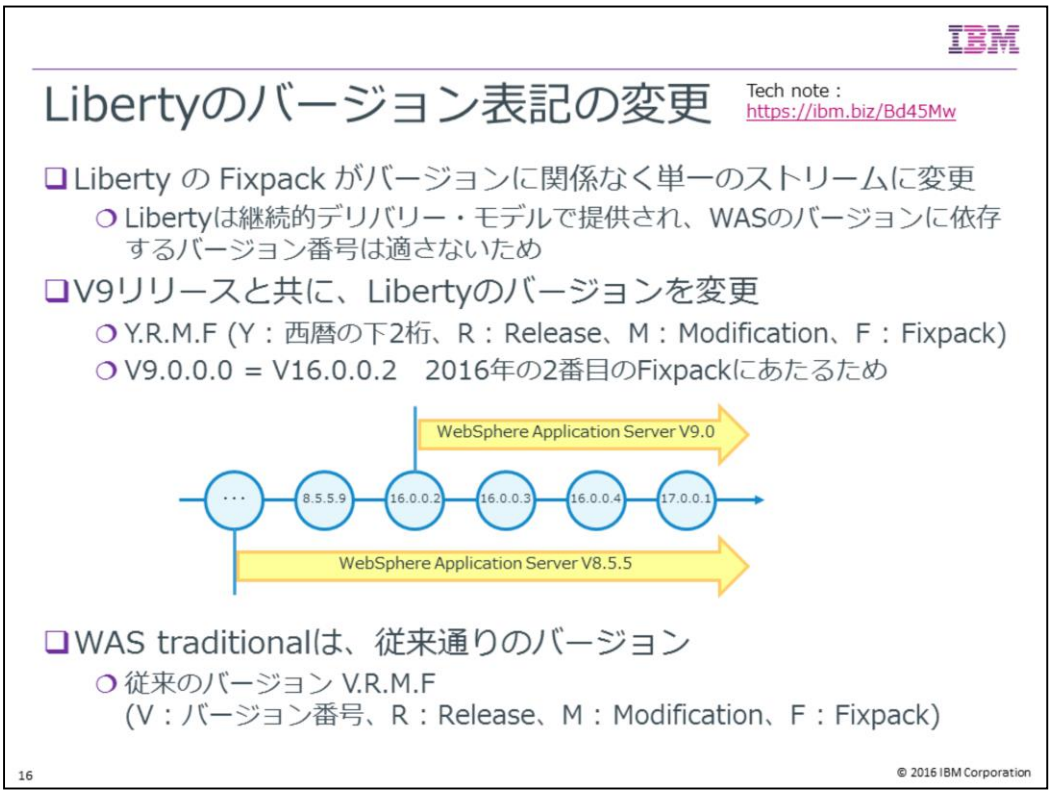
Libertyランタイムは、スピードが求められるSoEでの利用に適していますが、最近では、SoRでの利用も始まっています。一方、traditionalランタイムは、SoEの利用には適しておらず、SoRでの利用に適しています。



こちらは2014年末から、2016年6月までのLibertyの機能強化の抜粋です。

Libertyは、3ヵ月おきに新しい機能がリリースされています。2015年6月にJava EE 7に対応したバージョンがリリースされましたが、それ以前の2回は主に、Java EE 7の機能が追加されています。2015/9移行は、Java EE 仕様に含まれない拡張機能、管理機能、クラウド対応、API対応が行われてきています。

製品版リリースとは別に、製品版に組み込まれる前のLiberty Betaも提供されており、新しい機能が継続して追加されています。



Libertyの機能の継続的な提供を象徴する形として、バージョン表記が変更となりました。

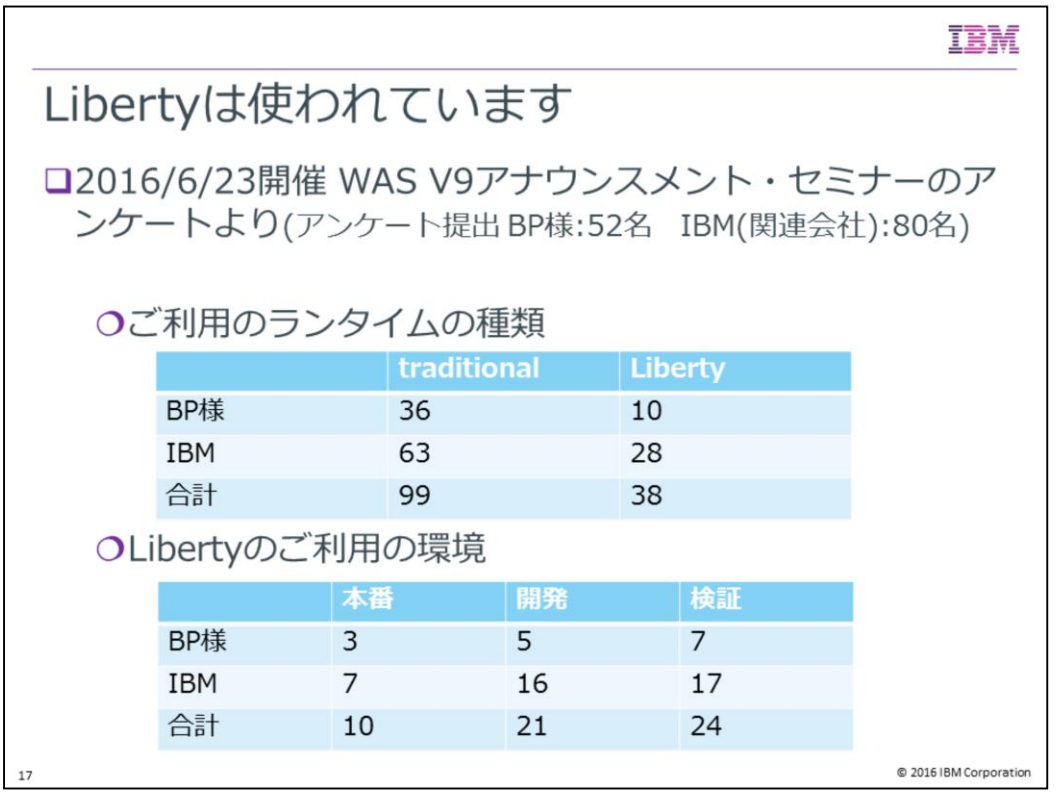
継続的デリバリー・モデルで提供されるため、これまでのバージョンに依存するバージョン番号は適さないための変更です。

バージョン表記は、Y.R.M.F (Y：西暦の下2桁、R：Release、M：Modification、F：Fixpack)となり、V9.0.0.0 = V16.0.0.2　になります。（2016年の2番目のFixpackにあたるため。）

Libertyは、今後この表記でバージョンが変更されていきます。

tWASに関しては、従来どおりのバージョン表記です。

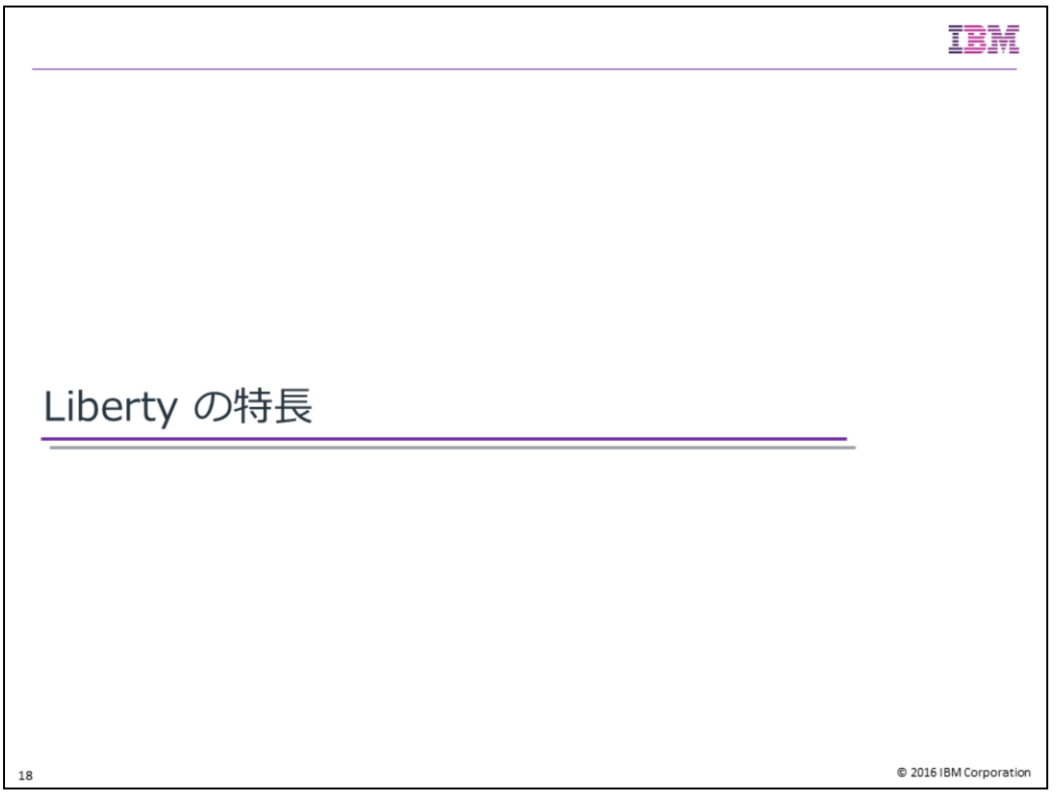
今後もLibertyの機能拡張、強化は継続して行われていきます。



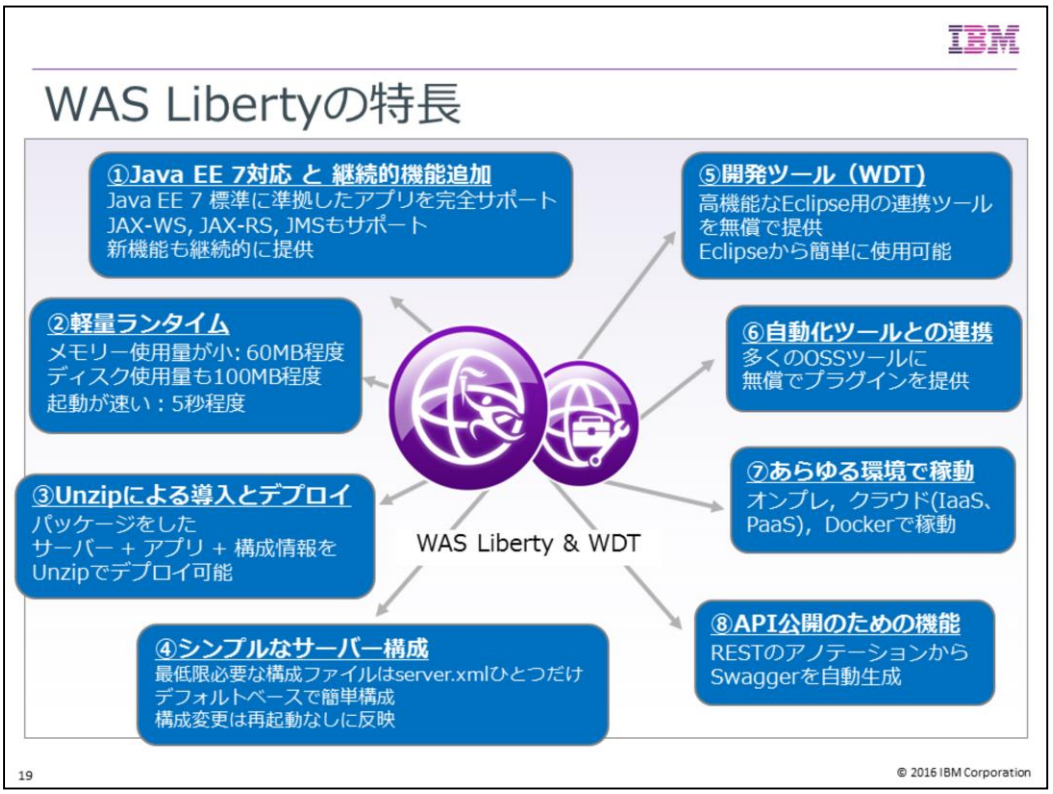
Liberty強化に伴い、Libertyの採用も増えてきています。

こちらは、2016/6/23に開催されたWAS V9アナウンスメント・セミナーのアンケート結果です。traditionalランタイムの方が、使用数としては多いですが、Libertyランタイムの利用もあることが確認できます。

また、Libertyランタイムの利用の中でも、本番での利用もされていることがわかります。Libertyの実績が増えてきており、今後の採用をご検討ください。



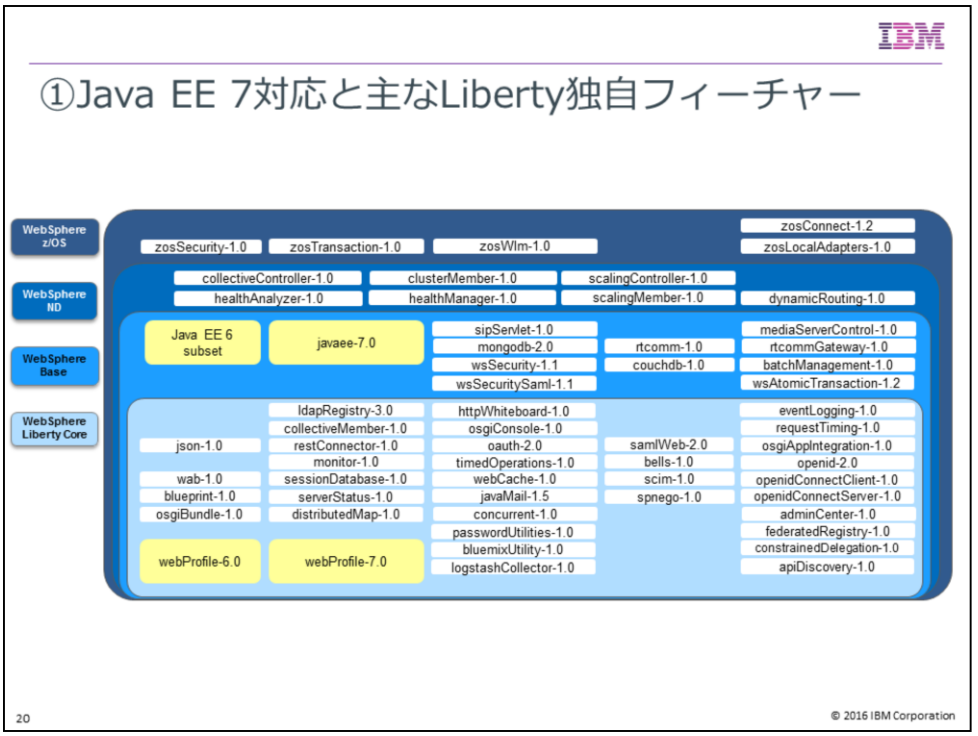
この章では、Libertyの特長についてご紹介します。



Libertyは、変化の激しいITに対応するための特長を持っています。
Libertyの代表的な特長は以下になります。

・Java EE 7対応、新機能の提供　→　①
・軽量高速、簡単構成　→　②③④
・ツール連携　→　⑤⑥
・クラウド対応　→　⑦
・API対応　→　⑧

以降のページでこれらの点について説明します。



こちらはLibertyのエディション毎に対応するフィーチャーを図示しています。

Java EE 仕様に関しては、薄い黄色で表しています。前述の通り、Liberty Coreエディションは、Web Profileのみのサポートになっており、Base以上でJava EE 7 Full Platformをサポートしています。

白枠のフィーチャーは、Libertyが提供する独自フィーチャーです。Java EE仕様以外にも多くの機能を提供していることが確認できます。NDエディションでは、拡張管理の機能として、collectiveのフィーチャーや、インテリジェント管理フィーチャーも確認できます。

こうした拡張機能が今後も追加されていく予定です。



Libertyは前述の通り継続的に新機能が追加されています。

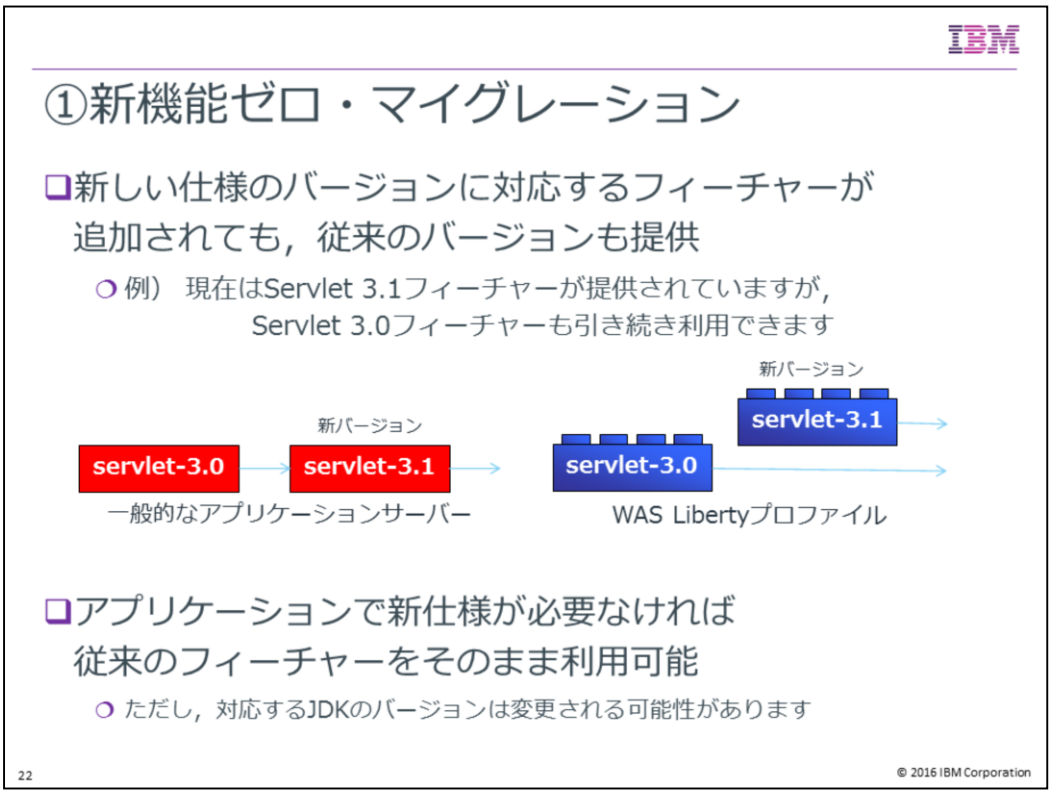
新機能は、まずBeta版に追加リリースされます。Beta版は、ここ2年程は1ヶ月間隔でリリースされています。

Betaとしてリリースされた機能は、その後、製品版としてリリースされます。製品版への新機能のリリースは、ここ2年ぐらいは3ヵ月間隔で行われています。

Betaへ追加された新機能や、製品版に追加された新機能の詳細は、WASdevのブログで確認することができます。

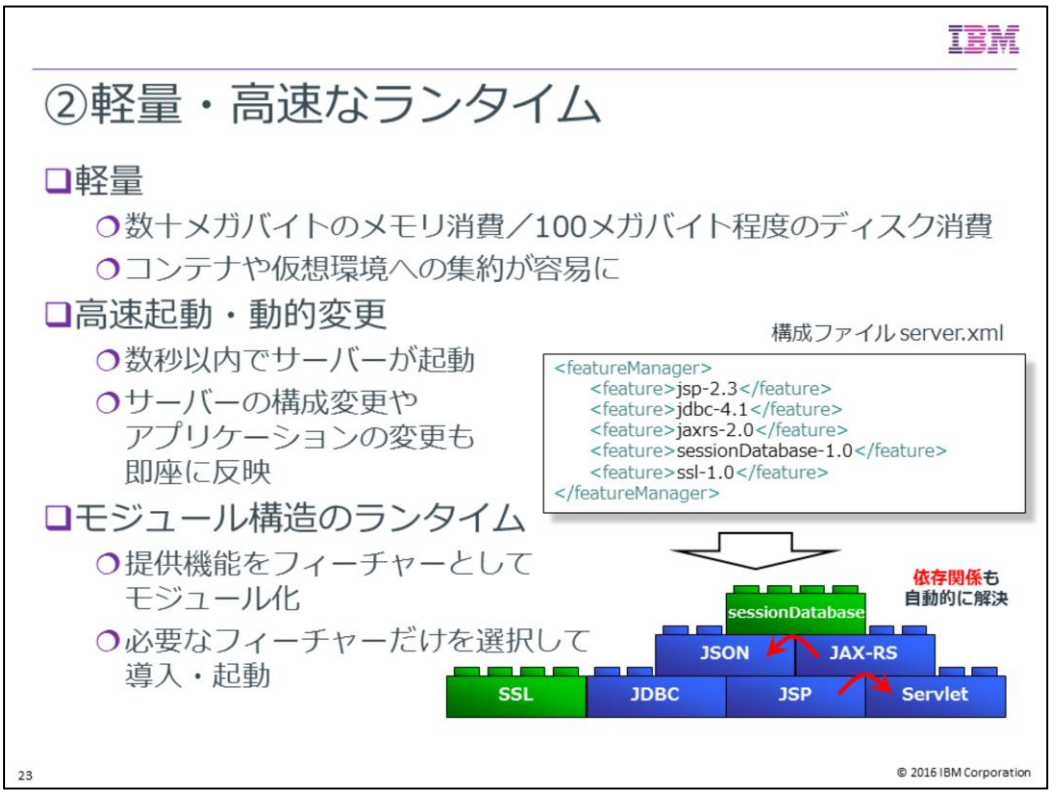
https://developer.ibm.com/wasdev/blog/category/blog/

これからも、このように最新のテクノロジーに対応できるよう継続的に機能追加が行われています。



Libertyは、ランタイムを置き換えるのではなく、フィーチャーを追加するというゼロ・マイグレーションという概念を持っています。新機能を利用したい場合には、新しいフィーチャー追加し、利用することができますし、必要なければ従来の機能を使い続けることができます。

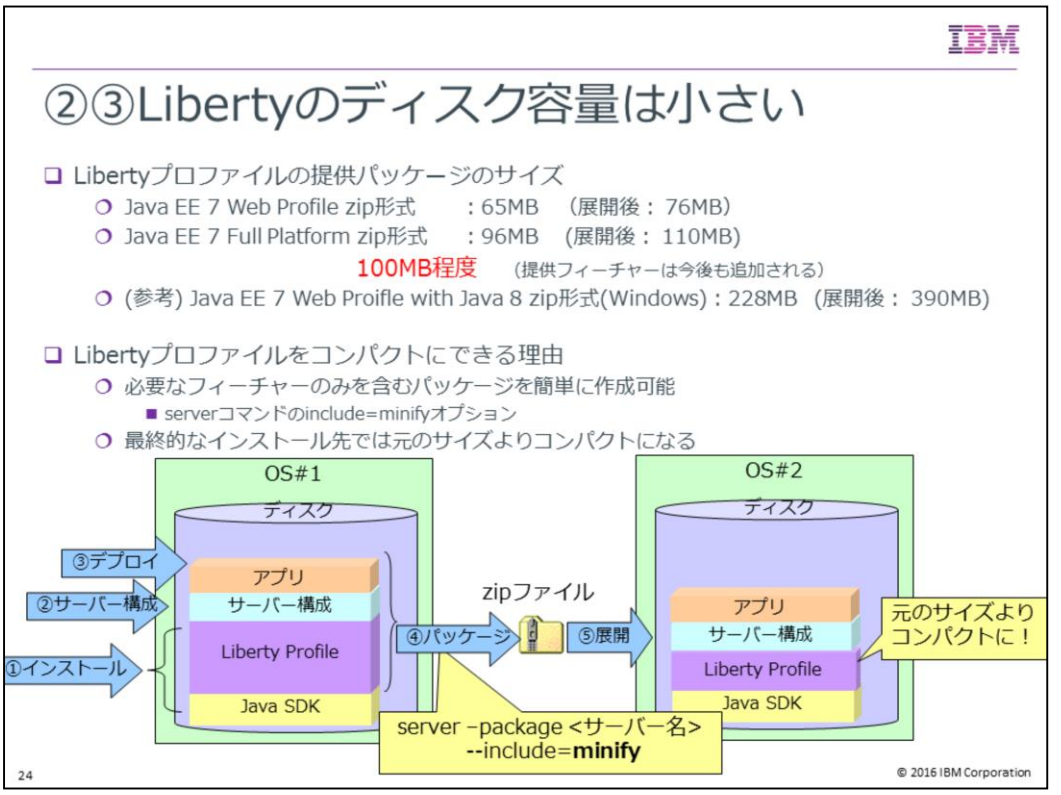
ゼロ・マイグレーションにより、新機能を使うために、これまでのように、ランタイムの更新(機能に対応したバージョンにバージョンアップ)を行い、アプリケーションも変更および全体のテストを行うといった労力をかける必要がありません。



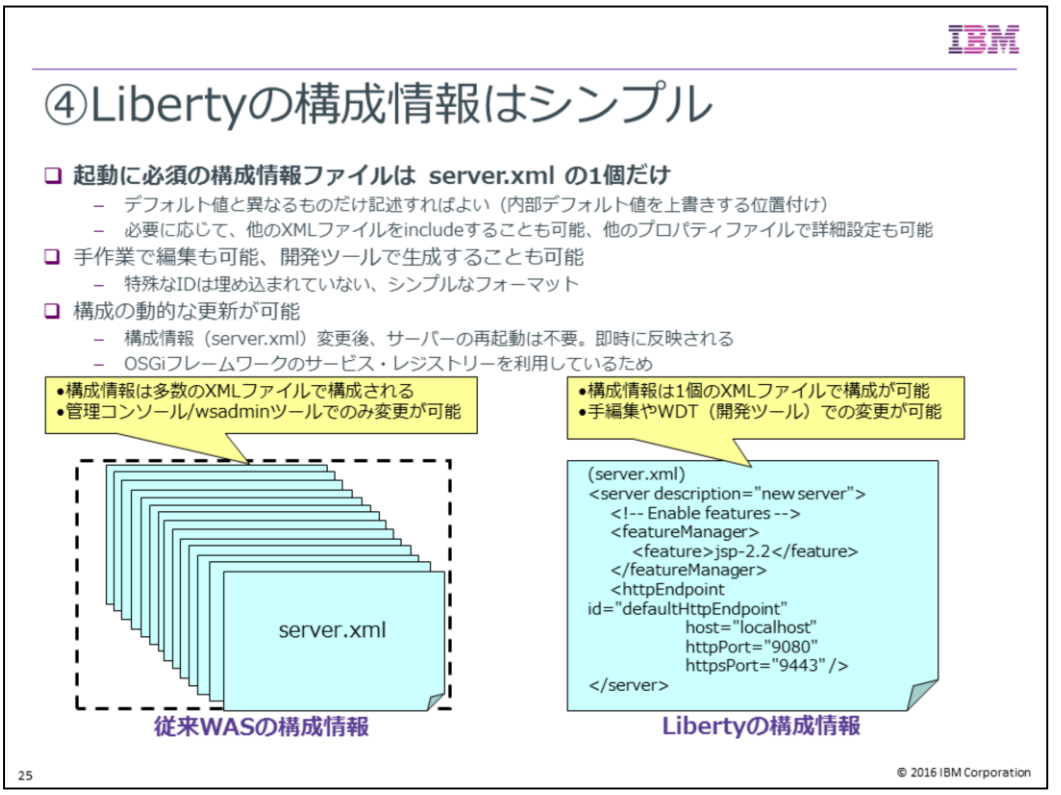
Libertyが軽量・高速な理由は、必要な機能単位（「フィーチャー」と呼びます）だけを柔軟に組み合わせて利用しているからです。

右の図がLibertyプロファイルでのフィーチャー利用のイメージ図です。Libertyのフィーチャーをブロックに喩えて描いています。Libertyでは、まるでブロックを積み上げるがごとく、必要なフィーチャーだけを組み合わせて利用できます。

使っていないフィーチャーはメモリーにロードされないため、高速に起動できますし、構成変更も簡単に即座に反映できます。



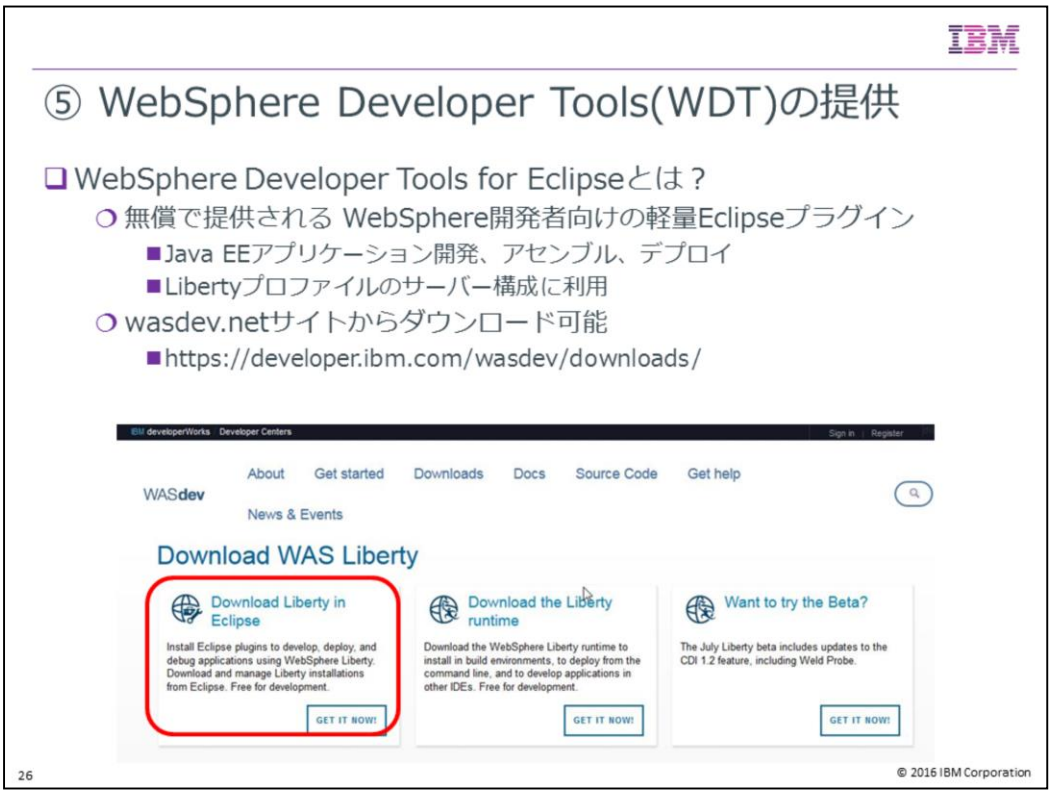
Libertyプロファイルが提供しているパッケージは、従来のWASと比べると非常に小さくなっています。提供されるフィーチャーは今後も追加されていく予定ですが、それでもLibertyのディスク使用量は小さく留めることが可能です。それは、Libertyの提供するserverコマンドでは、特定のサーバー構成で利用されているフィーチャーのみをパッケージするオプションが用意されているからです。そのオプションは、--include=minify　です。minifyは「小さくする」という意味です。



Libertyの構成情報は、最低限1個のXMLファイルで記述することが可能です。構成情報は、server.xmlという名前のファイルで、Libertyが内部に保持しているデフォルト値を上書きしたいものだけを記述すればよい仕組みになっています。右側に例として書かれているserver.xmlファイルは、わずか10行程度のものですが、これはデフォルトで生成されるserver.xmlファイルで、これだけでサーバーを起動することが可能です。もちろん、他のXMLファイルをインクルードしたり、詳細なオプションをプロパティー・ファイル（server.env, bootstrap.properties, jvm.options）で指定することも可能です。

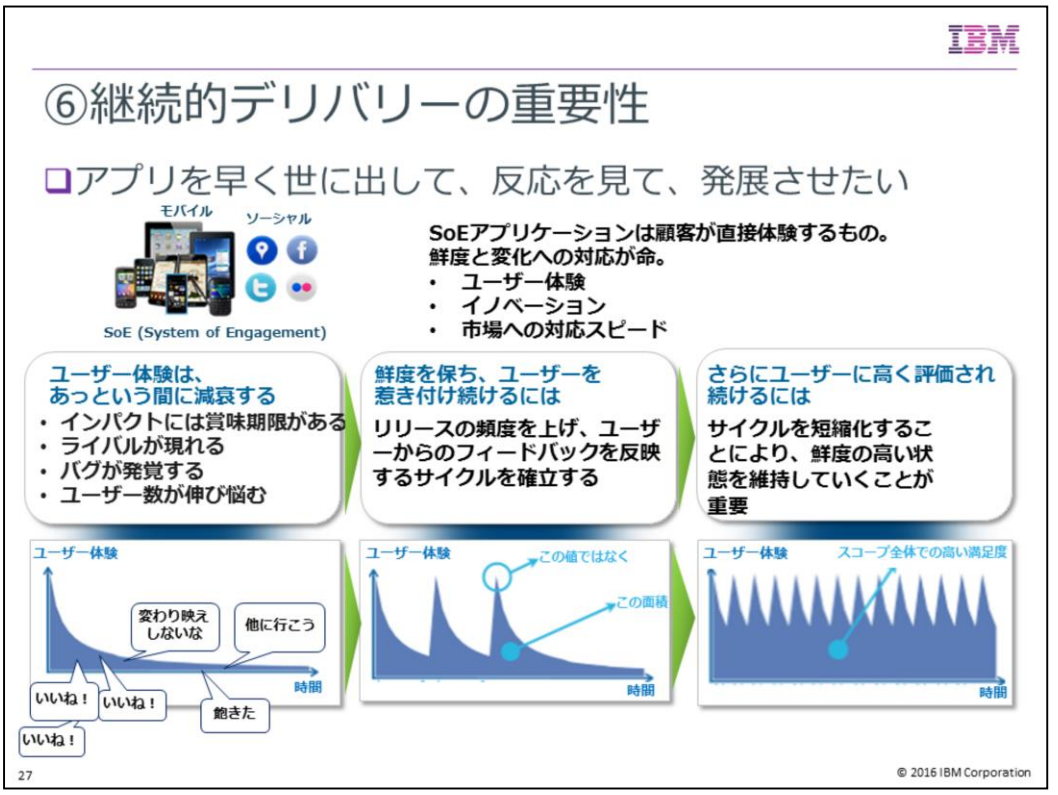
また、Libertyのserver.xmlファイルは、普通のエディターを利用して手作業で編集することもできますし、開発ツールのWDT (WebSphere Application Server Developer Tools for Eclipse) で編集することもできます。従来のWASでは、構成情報ファイルは40個以上のXMLファイルで構成され、手作業で編集することはできませんでした。これは、XMLファイルの中にWASが自動で生成する特殊なIDが埋め込まれていたからです。Libertyのserver.xmlファイルには、そのような特殊なIDは使われていません。

また、Libertyの構成情報ファイルの変更後は、その変更が即座にサーバー環境に反映されます。サーバーの再起動は不要です。



WebSphere Developer Toolsは、無償で提供されるWebSphere開発者向けの軽量Eclipseプラグインです。WebSphere Application Serverに向けて、Java EEの開発やアセンブルをヘルプする機能が提供されます。さらにLibertyプロファイルのサーバー構成をサポートする機能が豊富に含まれています。

WDTは、wasdev.netサイトからダウンロードすることが可能です。



SoEの領域では、ビジネスに与える価値を素早く判断し、継続して投資するのかやめるのか、ユーザーのフィードバックから軌道修正するのかということを行う必要があります。

たとえ、最初に良いアプリケーションをリリースできたとしても、図左のようにその体験はあっという間に減衰します。鮮度を保ち、ユーザーを繋ぎ止めて置くには、図真ん中のようにリリースの頻度を上げて、ユーザーからのフィードバックへの対応、より良いものを出し続けるサイクルを確立する必要があります。更に、ユーザーに高く評価され続けるには、図右のようにサイクルを短縮化して鮮度維持を行っていくことが重要です。

短いサイクルで継続してリリースする仕組みをいかに構築するのかが非常に重要になってきています。

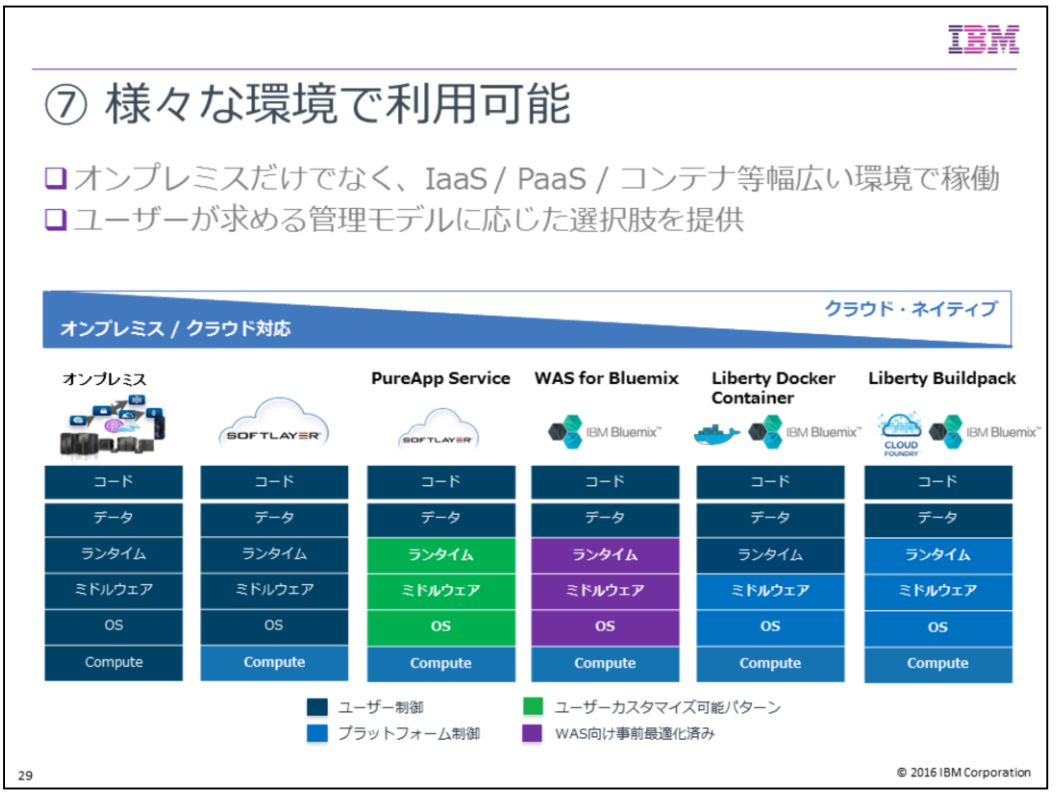


こうした継続的にリリースすることの重要性が上がっていることもあり、Libertyは、各種自動化ツールと連携するプラグインを提供しています。

IBMの継続的デリバリをサポートするツールであるUrbanCodeと連携するプラグインを提供しています。

OSSとの連携機能は、それぞれのOSSツールに対して、GitHub上でプラグインを公開しています。

Libertyと連携して仕様することで、システム開発のライフサイクルを自動化することが可能です。



前述の通り、Libertyは様々な環境で動作することができます。

Libertyは、オンプレミス、オフプレミス問わず様々な環境で動作させることができます。

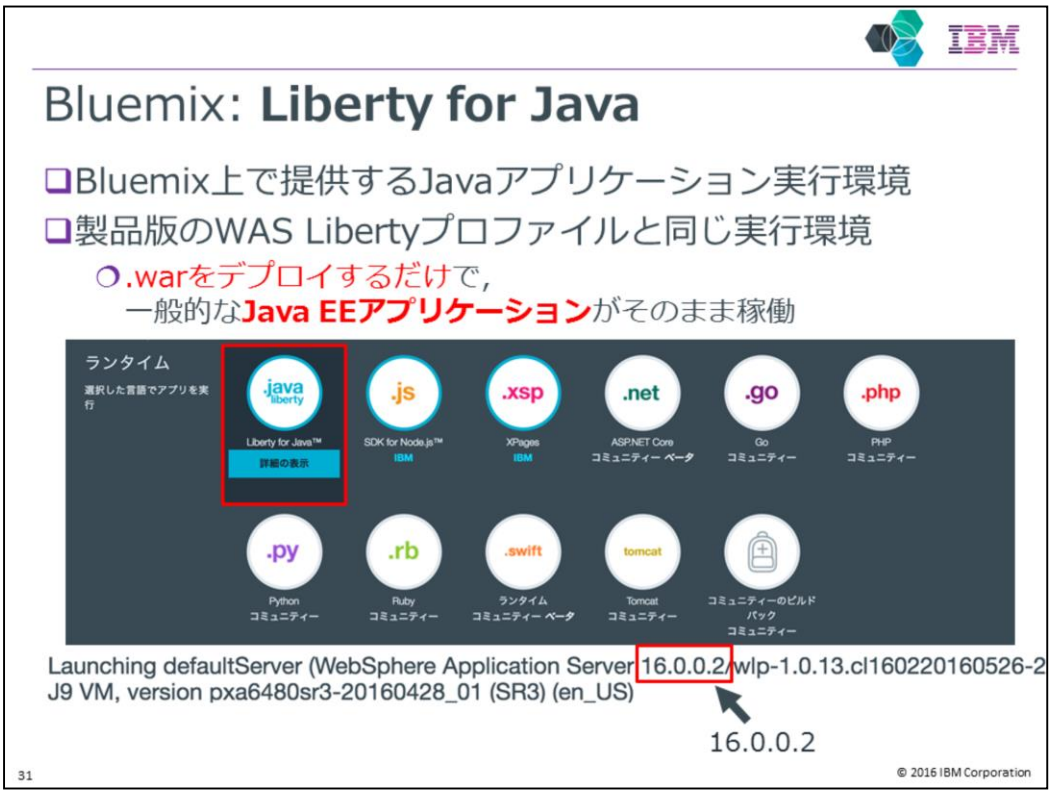
SoftLayerに代表されるIaaSの環境でのWASの稼動も可能です。PureApp Serviceは、PureApplicationサービスで培われたパターンの技術を利用できるものです。OS設定、クラスター環境、WAS設定をパターン化しておくことで、何度も同質の環境を作成することができます。

WAS for Bluemix、Liberty Docker、Liberty Buildpackについては、次ページ以降で説明します。



様々な環境でLibertyは利用できますが、Bluemix上でも、以下の3つのLibertyを提供しています。

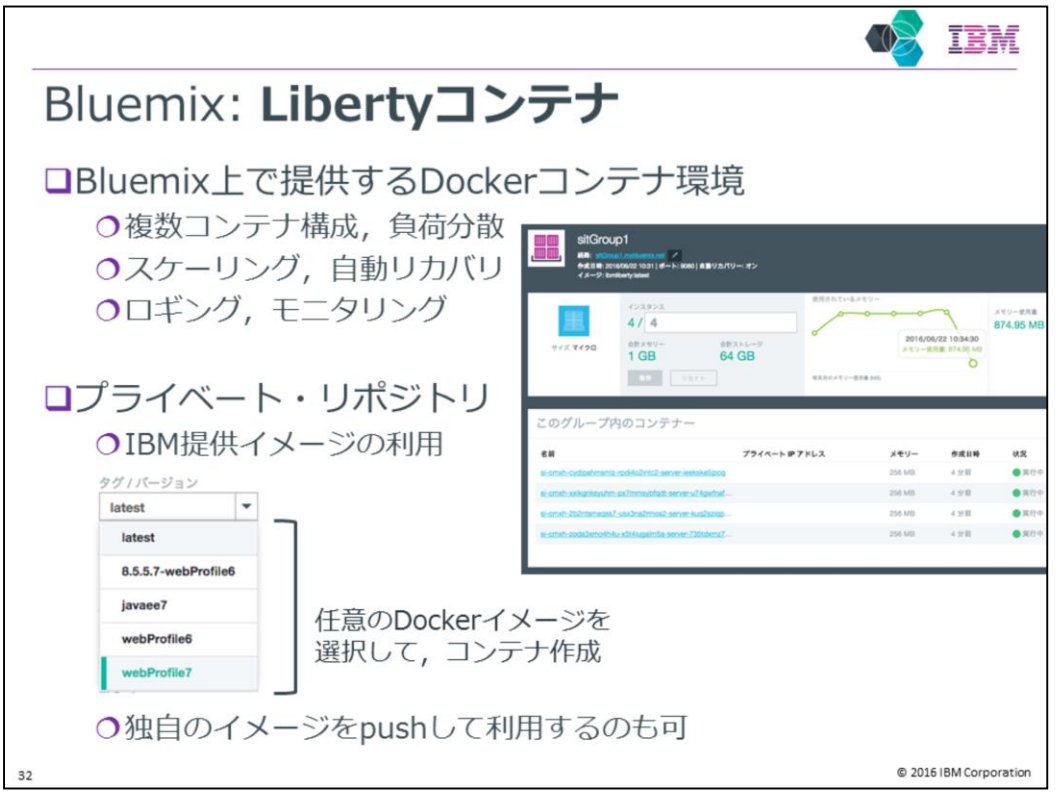
・Liberty for Java(Liberty Buildpack)
・Liberty コンテナ
・WAS for Bluemix



Liberty for Javaは、Bluemixで提供するJavaアプリケーションの実行環境です。

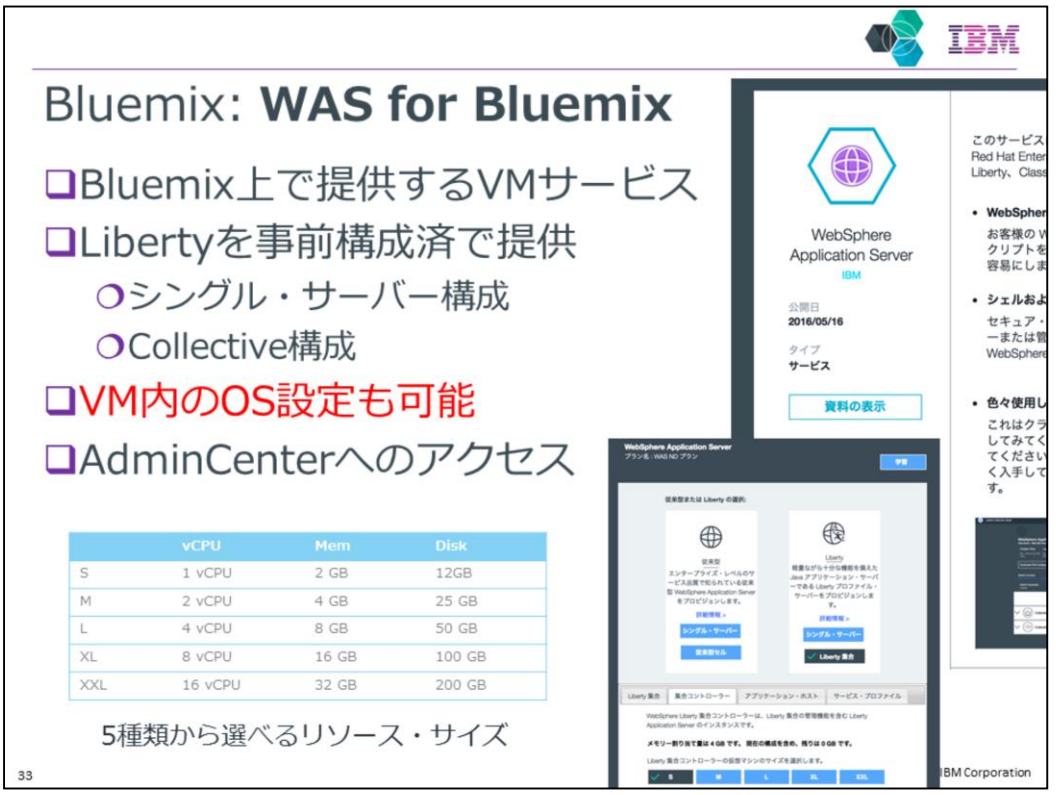
製品版のWAS Libertyプロファイルと同じ実行環境です。CloudFoundryベースのPaaS環境のためユーザーは、.warファイルをデプロイするだけで、一般的なJava EEアプリケーションを稼動させることができます。

Liberty for Javaは、Bluemixのメンテナンスにより最新のLibertyが提供されています。



Libertyコンテナは、Bluemix上で提供されているDockerコンテナ環境です。複数コンテナの構成による負荷分散やスケーリングにも対応しています。

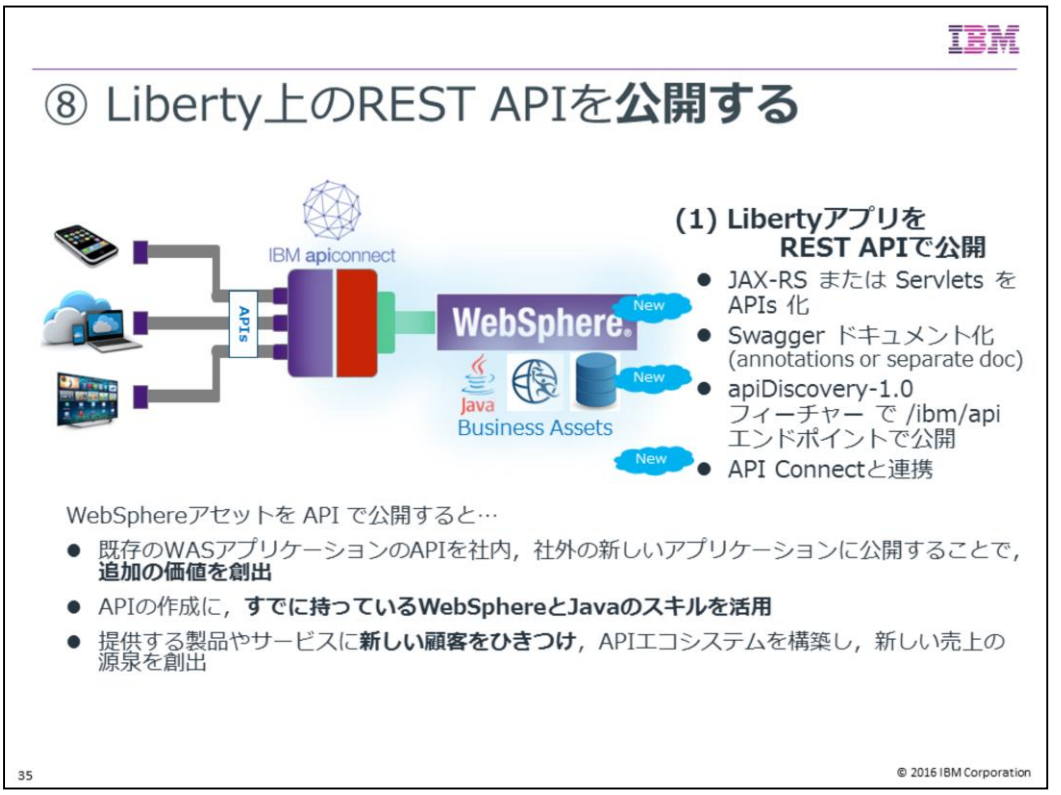
また、プライベート・リポジトリも提供しており、独自のDockerイメージを利用することもできます。



WAS for Bluemixは、Bluemix上で提供されるVirtual Machineのサービスです。WAS for Bluemixは前の2つと異なり、traditionalランタイムも構成することができます。WAS for Bluemixは、5種類のリソース・サイズから、Virtual Machineのサイズを選択できます。Libertyは、シングル・サーバー構成と複数サーバー構成のCollective構成をとることができます。また、Virtual Machineのため、OSにログインし、OSの設定や追加アプリケーションの導入を行うこともできます。また、Libertyの管理ツールであるAdminCenterへのアクセスも可能です。



先ほどのDocker コンテナはBluemix上のものでしたが、Dockerイメージも DockerHubに公開しています。このイメージを利用すると、他のDocker環境でも利用することができます。



最後の特長として、LibertyのAPI機能をご紹介します。

WASは、Java EE の仕様に含まれるJAX-RSを使用しREST APIのアプリケーションを作成することができます。これに加えてLibertyでは、API Discoveryフィーチャーを提供しています。API Discoveryでは、デプロイされたアプリケーションから、RESTのAnnotationを判別し、Swagger文書を作成し、公開します。また、公開されているREST APIをテストするSwagger UIも提供されます。

こうした機能を利用することで、既存のJavaスキル、既存のWASアプリケーションを活用し、API公開が可能となります。また、Swagger文書を利用することで、API利用者にインターフェース仕様を公開することもできます。



Libertyから他のサービスを呼び出す機能も提供されています。

bluemixUtilityでは、Bluemix上のサービスを検索し、インポートするコマンドを提供しています。クライアント・ライブラリのダウンロード、サーバー構成のバインドが可能です。

まだ、呼び出せるサービスは少ないですが、今後も連携するサービスが追加されていく予定です。

革新的なサービスを呼び出し、Libertyのアプリケーションを、低コスト、低リスクで拡張していくことができます。

IBM

まとめ

- WASの最新動向 & Libertyとは
  - クラウド対応
  - API機能
  - Liberty
- Libertyの特長
  1. Java EE 7対応 と 継続的機能追加
  2. 軽量ランタイム
  3. Unzipによる導入とデプロイ
  4. シンプルなサーバー構成
  5. 開発ツール（WDT)
  6. 自動化ツールとの連携
  7. あらゆる環境で稼動
  8. API公開のための機能

© 2016 IBM Corporation

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。 本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。 パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、WebSphereは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。
他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。
現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。

Windowsは Microsoft Corporationの米国およびその他の国における商標です。
JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会社の米国およびその他の国における商標または登録商標です。

© 2016 IBM Corporation



WebSphere Application Server Liberty

© 2016 IBM Corporation