IBM

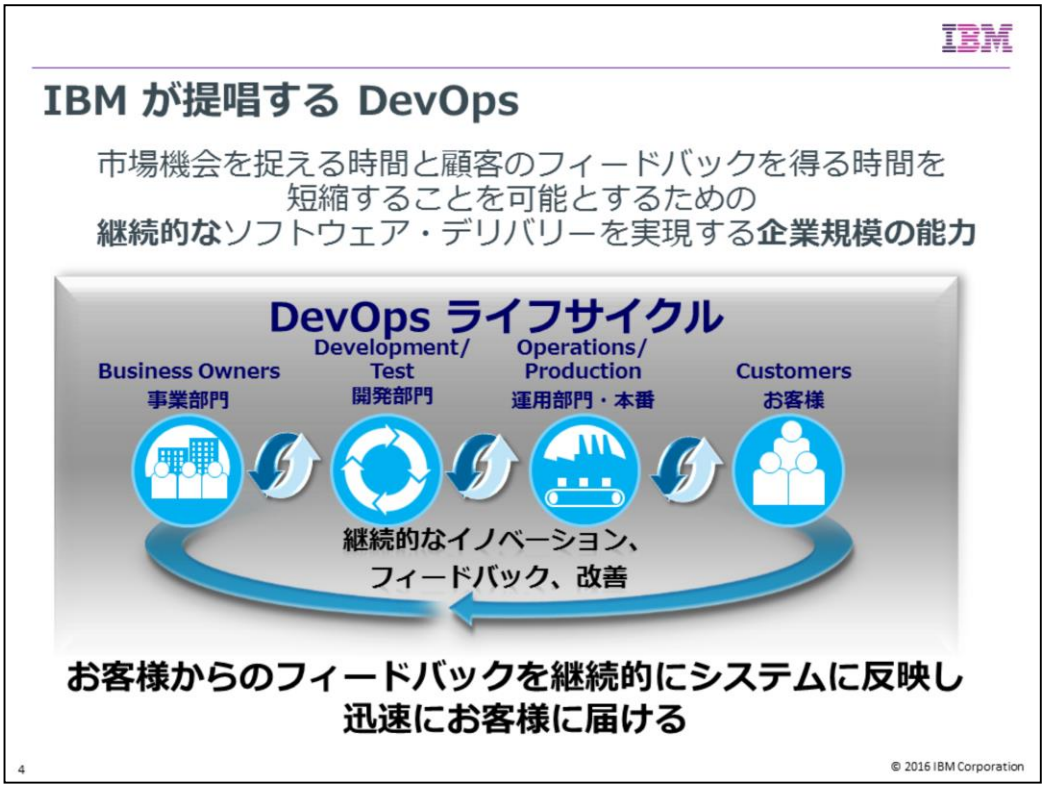
WebSphere Application Server V9

Liberty 基盤設計セミナー

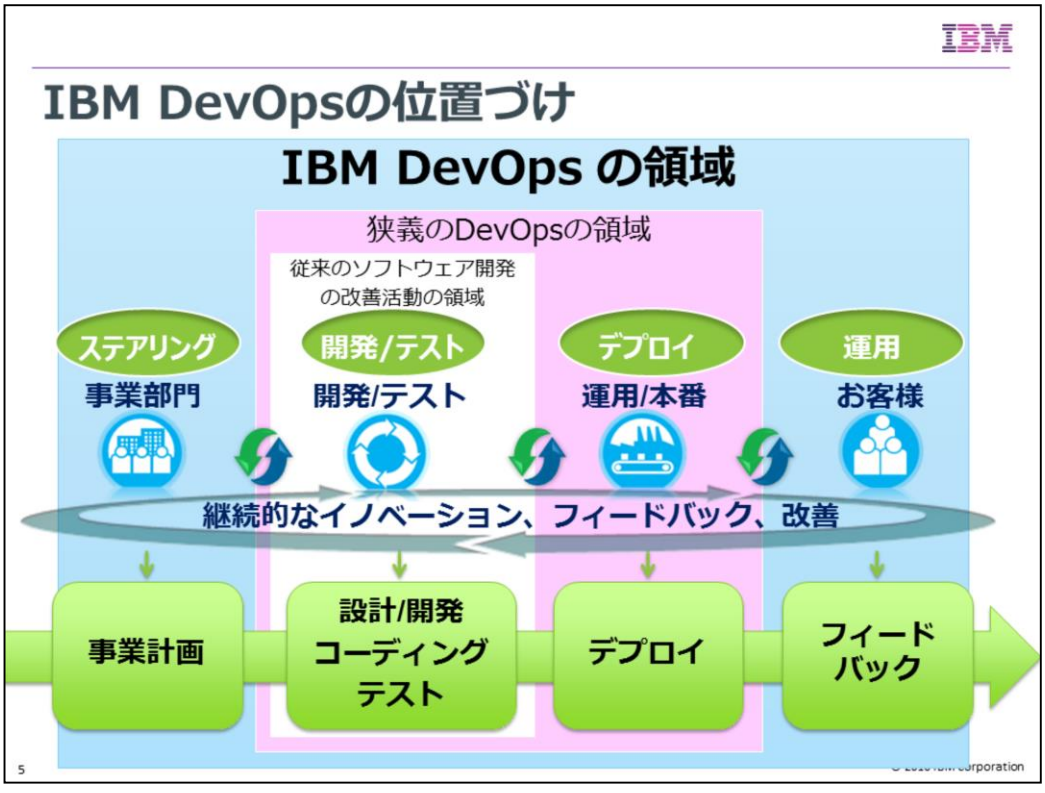
DevOpsツール連携

アジェンダです。関連する「運用設計」のセッションもご参照ください。

1. DevOps

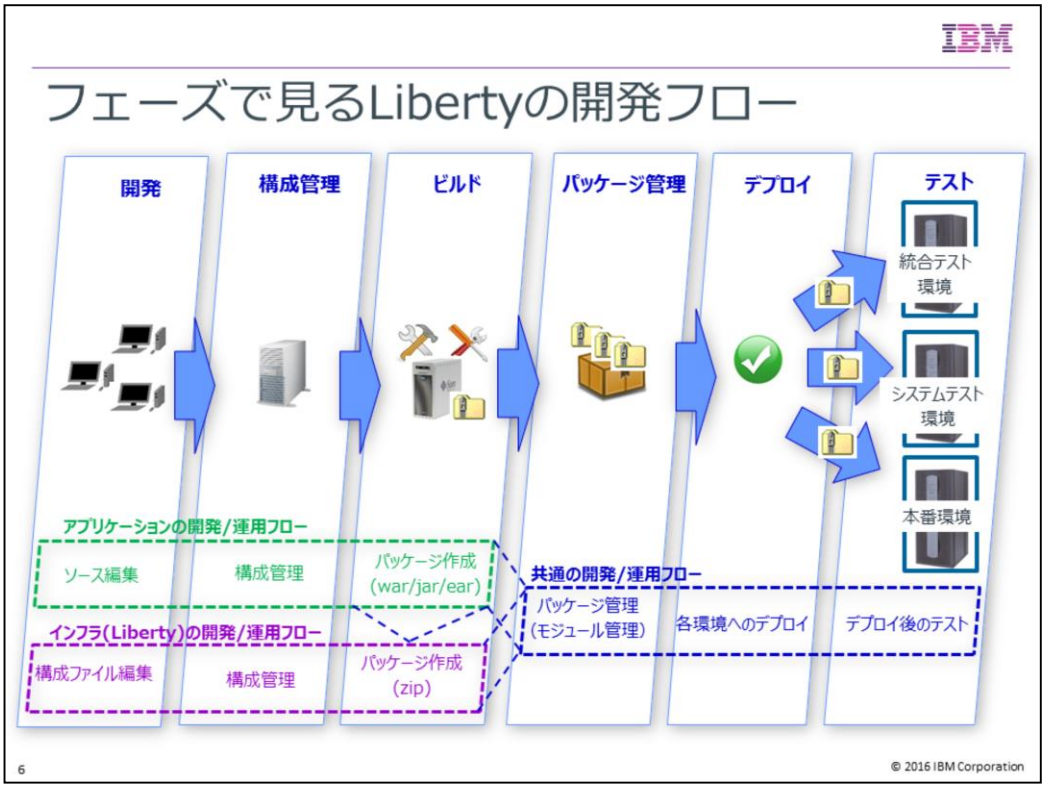


IBMが提唱しているDevOpsは、単にDevとOpsを融合して開発のスピードアップを図ることではなく、顧客の意見をフィードバックして開発に取り込み、それを安定した本番システムにきちんと反映し、さらに顧客からフィードバックを得てビジネス企画・開発に再び反映する取り組みと定義しています。

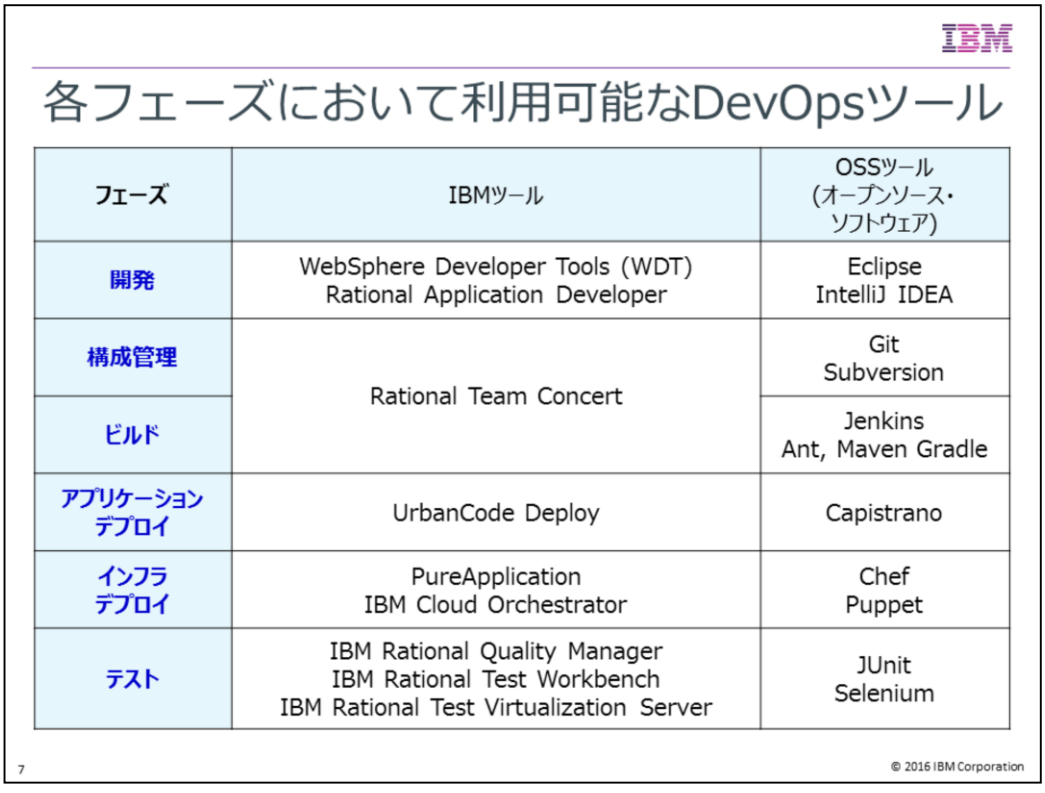


IBMが提唱しているDevOpsは、単にDevとOpsを融合して開発のスピードアップを図ることではなく、顧客の意見をフィードバックして開発に取り込み、それを安定した本番システムにきちんと反映し、さらに顧客からフィードバックを得てビジネス企画・開発に再び反映する取り組みと定義しています。

DevOpsのライフサイクルを回し、継続的なソフトウエアデリバリを実現することで、確実にビジネスの差別化につなげることがDevOpsの目的です。



Libertyの開発フローをフェーズ（開発工程）の流れとして見てみます。Libertyはアプリケーションだけでなく、インフラ(Libertyの構成ファイルなど）も開発対象となります。フェーズには大きく分けて「開発」、「構成管理」、「ビルド」、「パッケージ管理」、「デプロイ」、「テスト」に分かれます。各フェーズで行う作業をツール(DevOpsツール)で効率化することが開発生産性と品質を向上させるポイントです。次のページに主なDevOpsツールを紹介します。



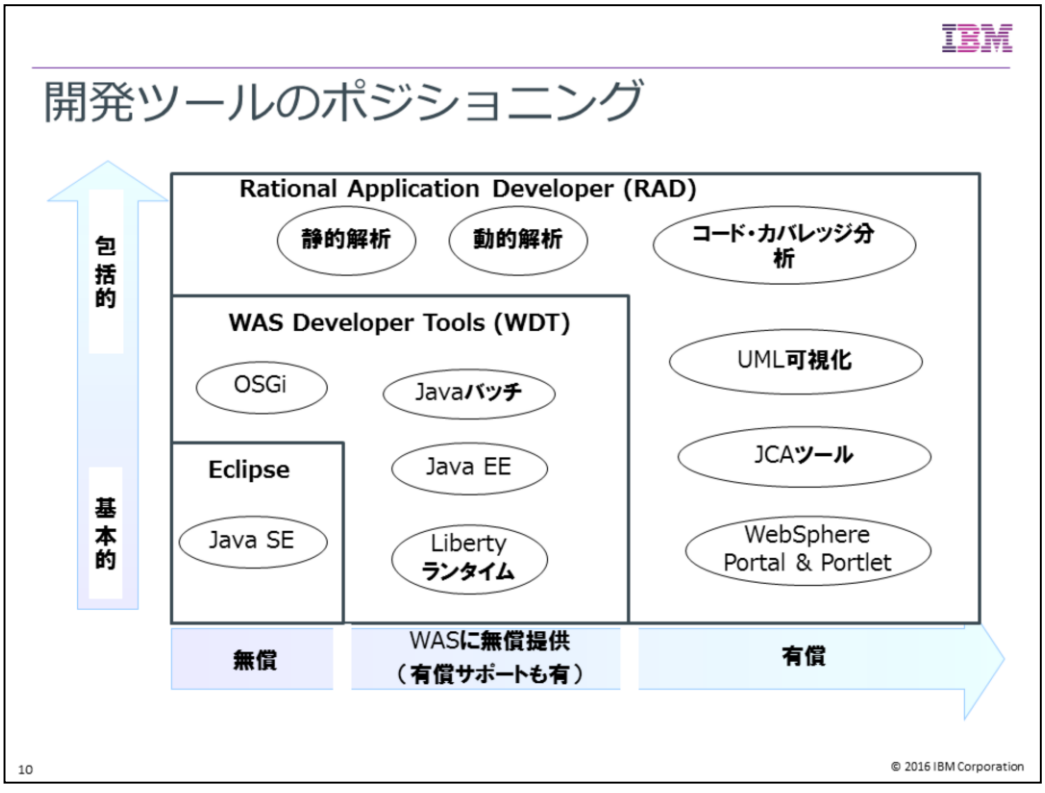
各フェーズにおいて利用可能な主なDevOpsツールを記載します。有償ツール(IBMツール）だけでなく、OSS(オープンソース・ソフトウェア)ツールも選択可能です。必要に応じて検討ください。

WASdevのDocsにはDevOpsのカテゴリーで、DevOpsツールとLibertyの連携に関する情報がまとめられています。

https://developer.ibm.com/wasdev/docs/category/devops/

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 Deploy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と連携して使用することで、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。

2. 開発ツール(WDT)



Libertyの開発ツールは無償/有償、機能の豊富さになどにより選択します。

WebSphere Developer Tools (WDT) はWAS 向けの Java EE アプリケーションを開発・アセンブルするための軽量のツール・セットで、Eclipse のアドオン・ツールです。Traditional WASとLibertyの両方に対応しており、WASの契約保守をお持ちのお客様に無償提供されます。次のページからWDTの紹介をします。

Rational Application Developer (RAD)はその上位となっており、WDTに加えて「静的解析」、「動的解析」、「コード・カバレッジ分析」、「UML可視化」、「JCAツール」、「WebSphere Portal開発」などの更なる開発生産性や品質向上のための機能を提供します。

WDT は、WAS 向けの Java EE アプリケーションを開発・アセンブルするための軽量のツール・セットで、Eclipse のアドオン・ツールです。WDT が提供する機能は大きく 2 つに分類できます。

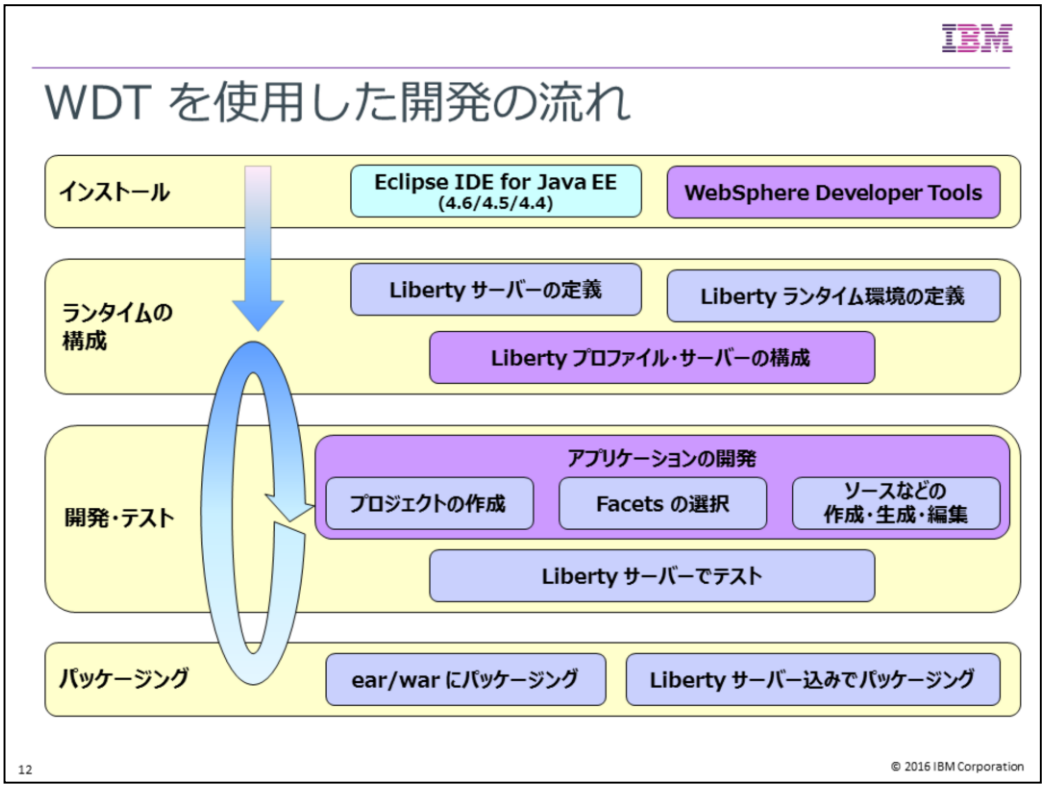
1 つめは、Liberty プロファイル・サーバーの構成機能です。たとえば、WDT が提供する Liberty Profile Configuration Editor を使用すると、Liberty サーバーの構成ファイルである server.xml ファイルをビジュアルに編集できます。bootstrap.properties, jvm.options, server.env などのファイルも作成・編集できます。また、アプリケーションを含めた状態で、Liberty プロファイル・サーバーを zip ファイルなどにパッケージングすることもできます。

2 つめは、 Java EE 7 アプリケーションの開発機能です。アプリケーションを構成する様々なファイルを作成・編集するために、各種のエディター、ウィザード、ツールなどが提供されており、Java EE アプリケーションの開発をスムーズに進めることができます。

尚、WDT が提供する Liberty プロファイル・サーバーの構成機能は、アプリケーションの開発だけではなく、Liberty プロファイル・サーバーの汎用的な構成ツールとして使用することができます。

WDTのシステム要件はこちらをご確認ください。

http://www.ibm.com/support/knowledgecenter/ja/SSHR6W_16.0.0/com.ibm.websphere.wdt.doc/topics/r_wdt_reqs.htm

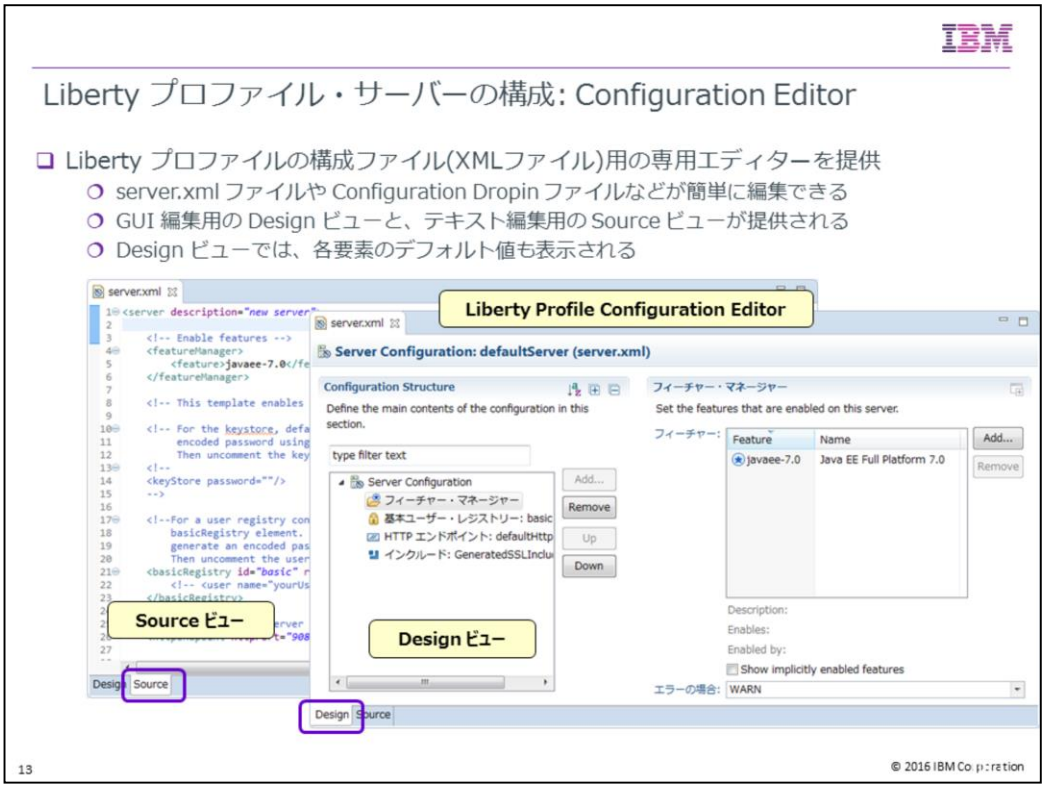


この図は、WDT を使用した開発の流れを表したものです。

最初のステップは、Eclipse と WDT のインストールで、次のステップがランタイムの構成です。ランタイムの構成では、開発・テストに使用する Liberty サーバーのサーバー定義を Eclipse に追加します。この時、Liberty プロファイルのインストール先や使用する JRE を指定し、Liberty サーバーのランタイム環境を定義し、Liberty サーバーを作成します。ランタイムの構成では、Liberty プロファイル・サーバーの構成も行います。

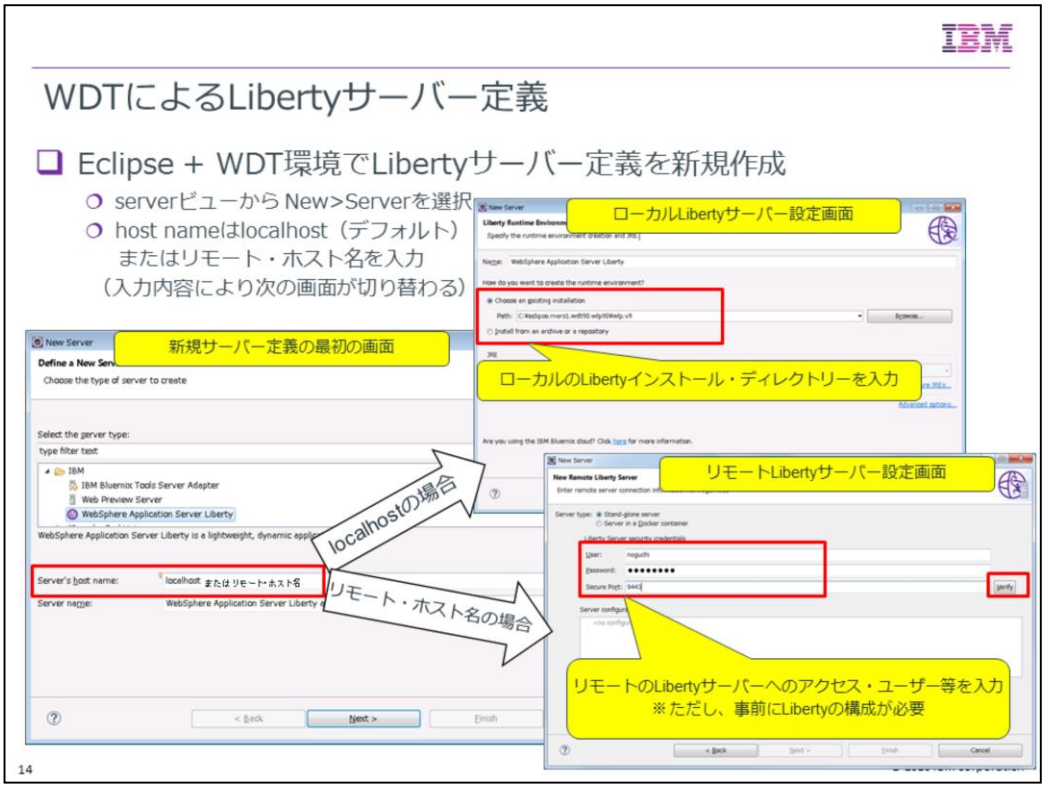
次のステップは、アプリケーションの開発・テストです。WDT が提供する各種エディター、ウィザード、ツールを使用してアプリケーションを開発し、前のステップで作成した Liberty サーバーにデプロイしてテストを行います。

開発・テストが完了したアプリケーションは、WDT の機能を使用して、ear ファイルや war ファイルにエクスポートしたり、Liberty サーバーのランタイムを含めた状態で zip ファイルなどにパッケージングすることができます。



上の図は、Liberty Profile Configuration Editor の画面です。

Liberty Profile Configuration Editor は、Liberty プロファイルの構成ファイル(XMLファイル)用の専用エディターで、「Servers」ビューで Server Configuration (server.xml) などをダブル・クリックすると表示されます。GUI 編集用の Design ビューと、テキスト編集用の Source ビューが提供されており、server.xml ファイルや Configuration Dropin ファイルなどの編集に使用できます。Design ビューでは、各要素のデフォルト値も表示されます。



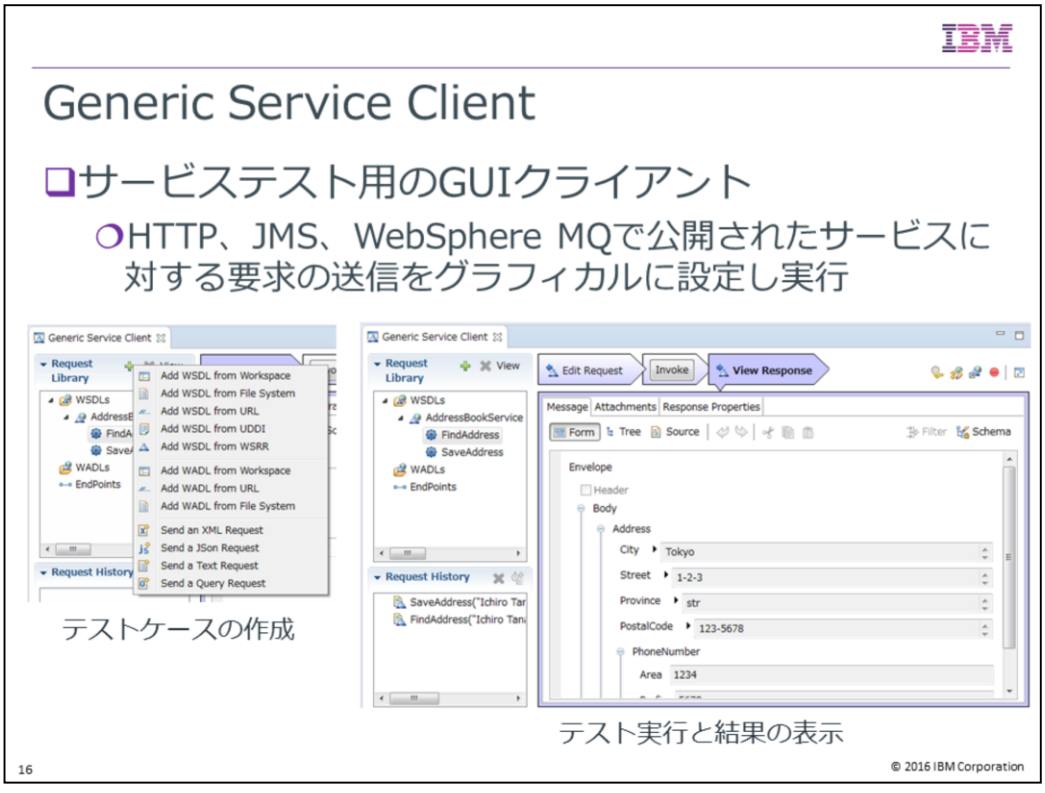
WDTでは従来からローカルのLibertyサーバーを操作することはできましが、比較的最近になって、リモートのLibertyサーバーにも対応しました。

チャートではその設定方法を示しています。新規サーバー定義の画面で、「Server's host name」に「localhost」（デフォルト値）のまま次に進むと、ローカル構成の画面が表示されます。このフィールドにリモート・ホスト名を入力して次に進むと、リモート構成の画面が表示されます。

リモートLiberty設定画面では、リモートにあるLibertyサーバーのアクセス情報として、ユーザー名、パスワード、ポート番号を入力して、「Verify」を押します。ただし、このWDT構成を行うためには、事前にリモートLibertyサーバーでの設定が必要です。詳細は「運用設計」の「構成変更」の章を参照してください。

WDT 16 から提供された主な新規は上記の通りです。アプリケーションの開発機能の拡張としては、Generic Service ClientとAPI開発機能が大きな拡張点です。

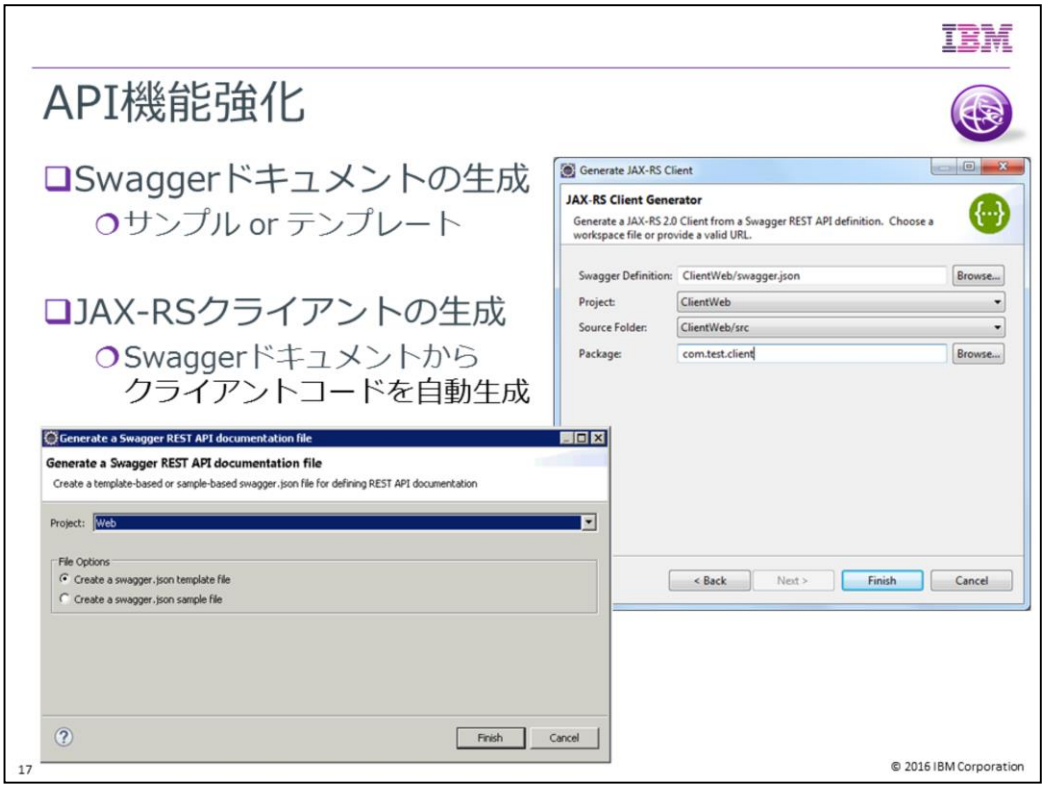
概要: IBMWebSphere Application Server Developer Tools for Eclipse
http://www.ibm.com/support/knowledgecenter/ja/SSHR6W_16.0.0/com.ibm.websphere.wdt.doc/topics/wdt_overview.htm



Generic Service Client（汎用サービス・クライアント）は、HTTP、JMS、またはWebSphere MQ トランスポートを使用するあらゆる種類のサービスの呼び出しを起動し、サービスから 戻されるメッセージを表示します。汎用サービス・クライアントは、サービスの呼び出しの起動を行う専用クライアントへのアクセス権限がない場合に、サービスのデバッグやテストを行う際に便利です。サービスのさまざまなトランスポートおよびセキュリティー構成をセットアップしたり、 呼び出しのパラメーターを編集したり、添付を送信したりすることができます。

詳細はオンライン・マニュアルの「汎用サービス・クライアントを使用した Web サービスのテストの概要」を参照してください。

http://www.ibm.com/support/knowledgecenter/ja/SSHR6W_16.0.0/com.ibm.websphere.wdt.doc/topics/core/cgenservclient.htm



LibertyはAPI機能が強化されていますが、WDTではAPI開発支援機能を提供します。例えば、JSON 定義ファイルを作成するための Swagger 2.0 構文 (JSON 構文に基づく) をサポートします。

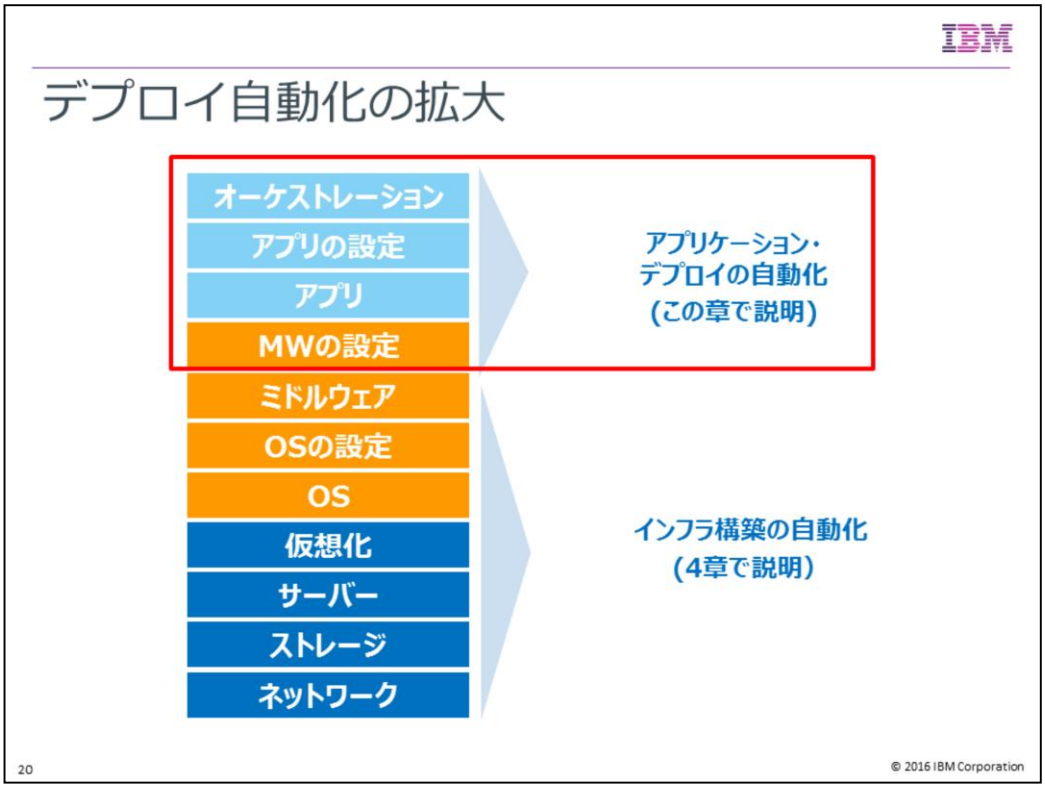
詳細はオンライン・マニュアルの「REST API 定義ファイルの開発」を参照してください。

http://www.ibm.com/support/knowledgecenter/ja/SSHR6W_16.0.0/com.ibm.websphere.wdt.doc/topics/core/cgenservclient.htm

IBM

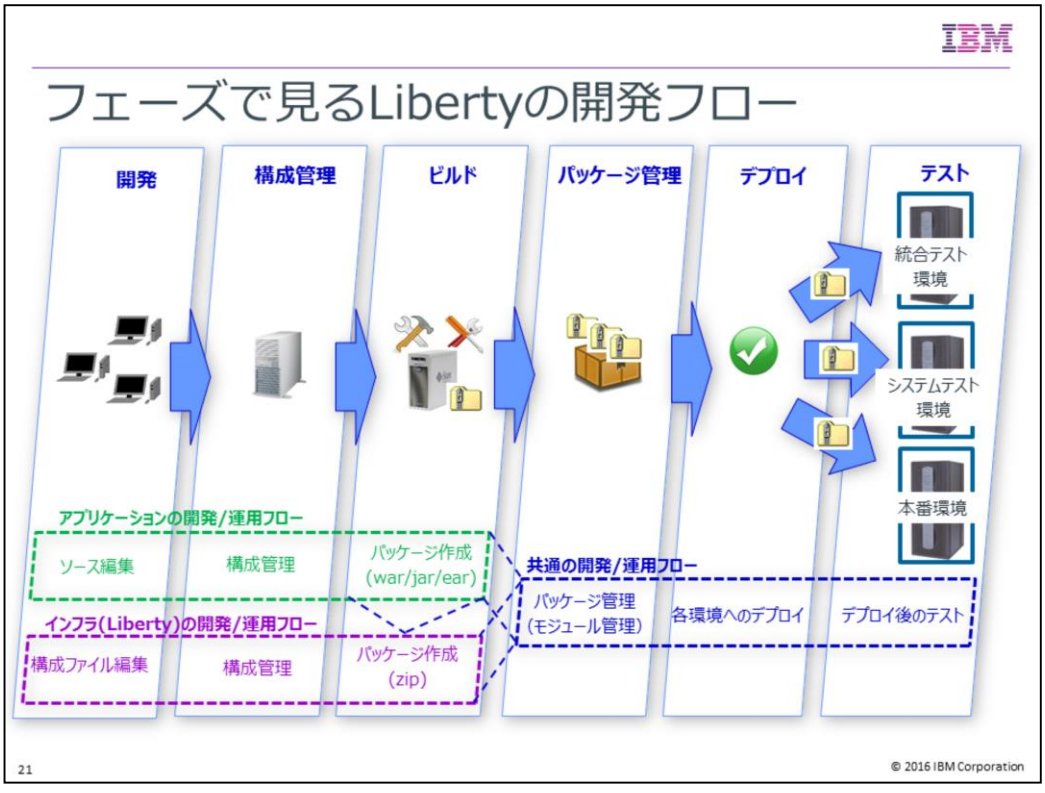
3. アプリケーション・デプロイの自動化

P7で紹介したDevOpsツールをどのフェーズ（開発工程）で利用するかを図にまとめました。各フェーズで行う作業をツール(DevOpsツール)で効率化することが開発生産性と品質を向上させるポイントです。



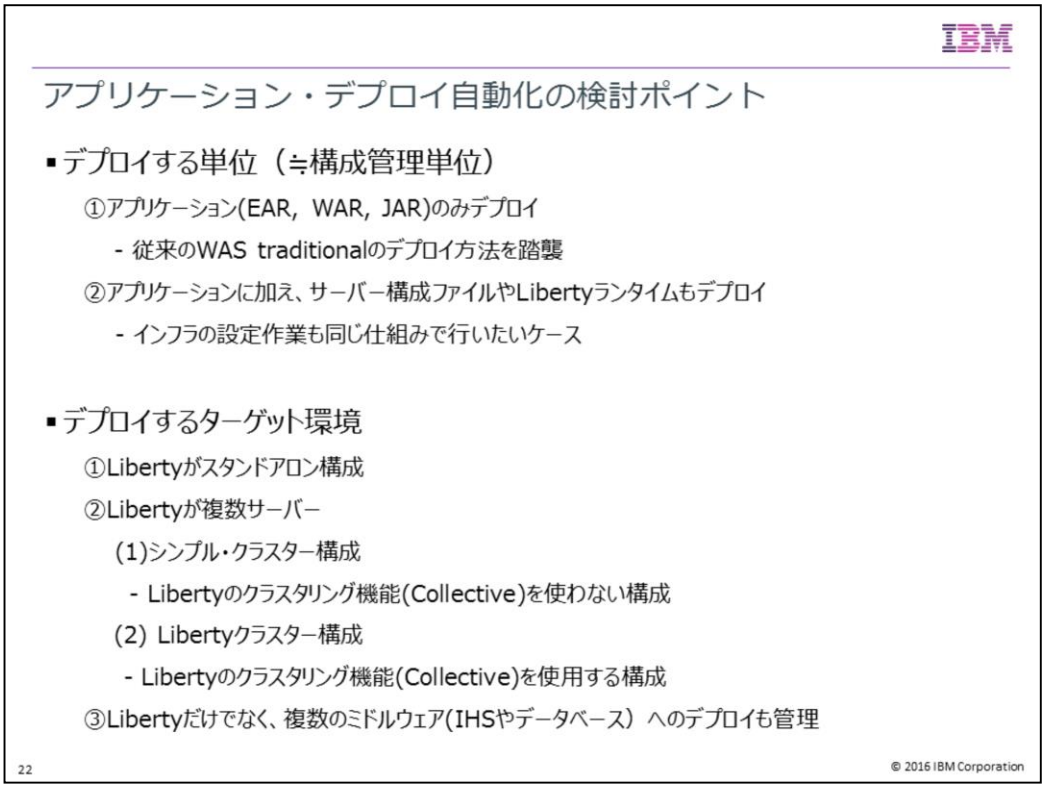
デプロイ自動化を考えた場合、大きく「アプリケーション・デプロイの自動化」と「インフラ構築の自動化」の２つに分けられます。

この章では「アプリケーション・デプロイの自動化」を紹介します。ここには
・オーケストレーション（複数アプリの同時デプロイや順序性の制御など）
・アプリの設定
・アプリ
・MW設定（Libertyの設定）
が含まれます。



ここで「フェーズで見るLibertyの開発フロー」のページを再掲します。Libertyはアプリケーションだけでなく、インフラ(Libertyの構成ファイルなど）も開発対象となります。

これらの作業は前のページの「アプリケーション・デプロイの自動化」に含まれます。



アプリケーション自動化の検討ポイントは大きく2つあります。

・デプロイする単位（これは構成管理単位にほぼ等しいと考えてください）
①アプリケーション(EAR, WAR, JAR)のみデプロイ
②アプリケーションに加え、サーバー構成ファイルやLibertyランタイムもデプロイ

・デプロイするターゲット環境

①Libertyがスタンドアロン構成

②Libertyが複数サーバー

(1)シンプル・クラスター構成

(2) Libertyクラスター構成

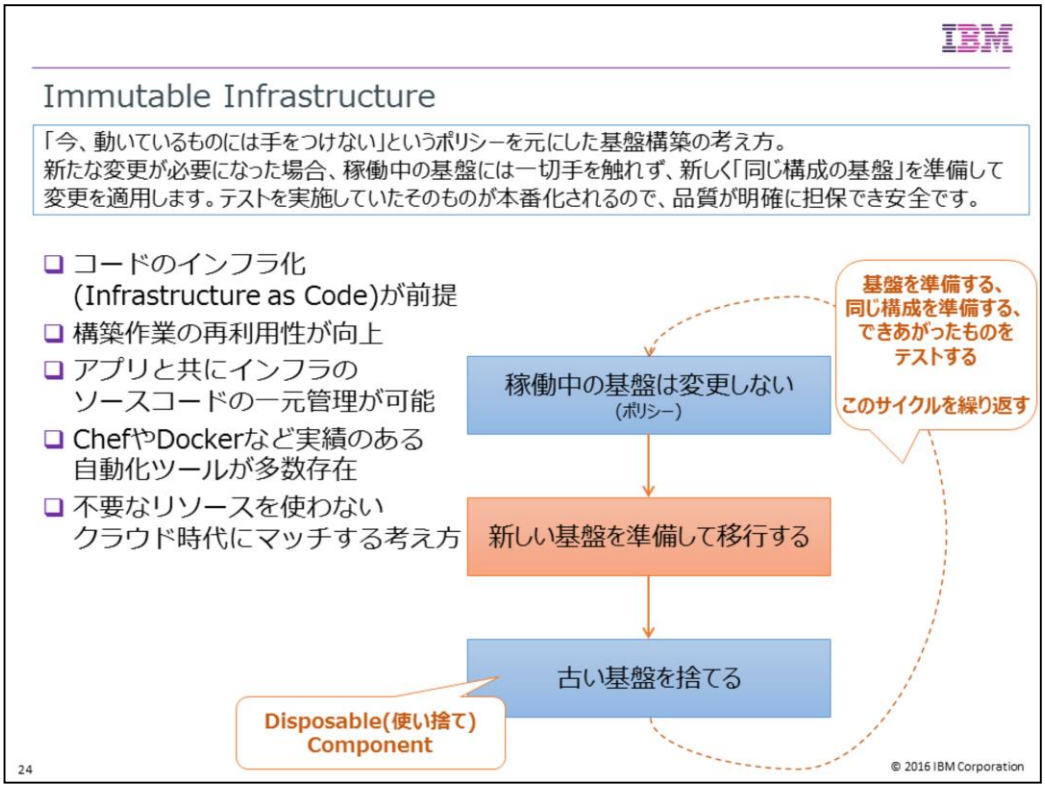
③Libertyだけでなく、複数のミドルウェア(IHSやデータベース）へのデプロイも管理

次のページから詳細を説明します。

「デプロイする単位」は、①アプリケーション(EAR, WAR, JAR)のみデプロイと、②アプリケーションに加え、サーバー構成やLibertyランタイムもデプロイの2通りが考えられます。従来のJava EEアプリケーションサーバー(WAS traditionalなど）は①を取ることが多いので、こちらの方が馴染みがあるかもしれません。違いはアプリケーションだけでなく、インフラ(Libertyの構成ファイルなど）も構成管理して、デプロイを同時にできるようにするかどうかになります。

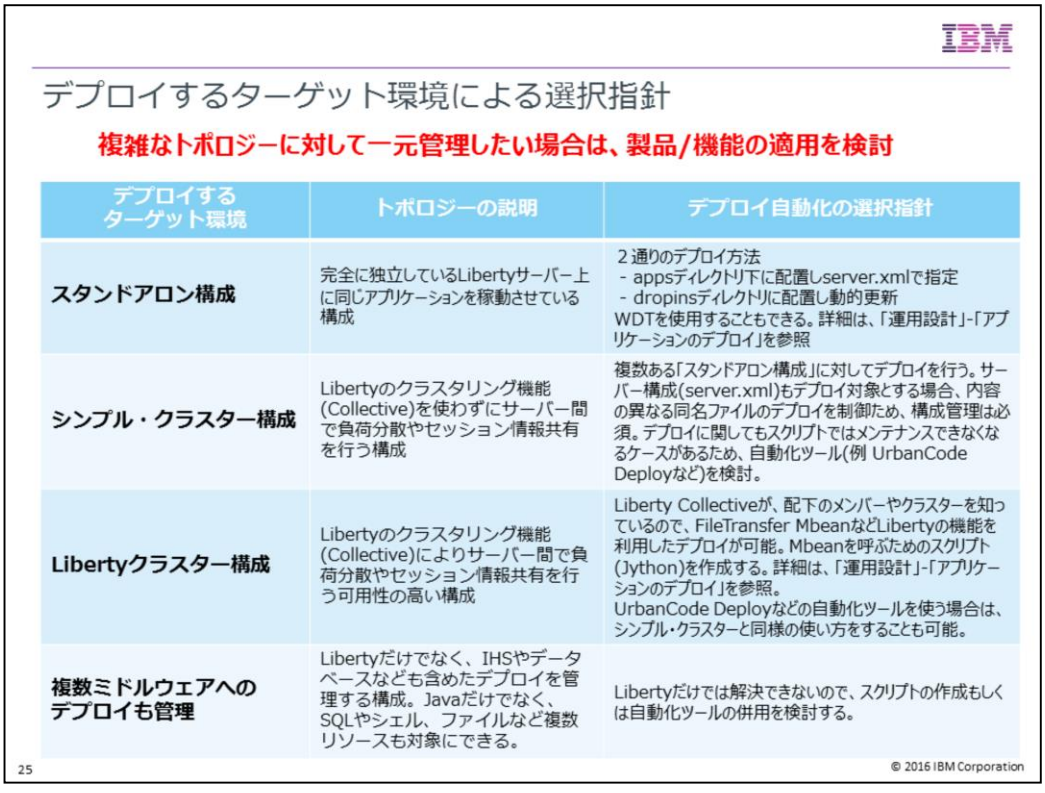
②は仮想化やクラウドを基盤とした新しいインフラの考え方である「Immutable Infrastructure」を実現することができますが、アプリチームとインフラチームの協業が必要です。プロジェクト組織（アプリ、インフラ）の要件に応じて、デプロイする単位を検討してください。

前のセッションの「運用設計」-「アプリケーションのデプロイ」も参照ください。



仮想化やクラウドを基盤とした新しいインフラの考え方である「Immutable Infrastructure」が注目されています。これは「今、動いているものには手をつけない」というポリシーを元にした基盤構築の考え方であり、新たな変更が必要になった場合、稼働中の基盤には一切手を触れず、新しく「同じ構成の基盤」を準備して変更を適用します。テストを実施していたそのものが本番化されるので、品質が明確に担保でき安全です。

これを実現するにはコードのインフラ化(Infrastructure as Code)が前提となります。前のページのようにLibertyはアプリと共にインフラ(Libertyの構成ファイルなど）も構成管理の対象となりますので、一元管理が可能です。4章で説明するインフラ構築の自動化ツールのChefやDockerと組み合わせることも可能です。



デプロイするターゲット環境による選択指針

複雑なトポロジーに対して一元管理したい場合は、製品/機能の適用を検討

デプロイするターゲット環境	トポロジーの説明	デプロイ自動化の選択指針
スタンドアロン構成	完全に独立しているLibertyサーバー上に同じアプリケーションを稼動させている構成	2通りのデプロイ方法 - appsディレクトリ下に配置しserver.xmlで指定 - dropinsディレクトリに配置し動的更新 WDTを使用することもできる。詳細は、「運用設計」-「アプリケーションのデプロイ」を参照
シンプル・クラスター構成	Libertyのクラスタリング機能(Collective)を使わずにサーバー間で負荷分散やセッション情報共有を行う構成	複数ある「スタンドアロン構成」に対してデプロイを行う。サーバー構成(server.xml)もデプロイ対象とする場合、内容の異なる同名ファイルのデプロイを制御ため、構成管理は必須。デプロイに関してもスクリプトではメンテナンスできなくなるケースがあるため、自動化ツール(例 UrbanCode Deployなど)を検討。
Libertyクラスター構成	Libertyのクラスタリング機能(Collective)によりサーバー間で負荷分散やセッション情報共有を行う可用性の高い構成	Liberty Collectiveが、配下のメンバーやクラスターを知っているので、FileTransfer MbeanなどLibertyの機能を利用したデプロイが可能。Mbeanを呼ぶためのスクリプト(Jython)を作成する。詳細は、「運用設計」-「アプリケーションのデプロイ」を参照。 UrbanCode Deployなどの自動化ツールを使う場合は、シンプル・クラスターと同様の使い方をすることも可能。
複数ミドルウェアへのデプロイも管理	Libertyだけでなく、IHSやデータベースなども含めたデプロイを管理する構成。Javaだけでなく、SQLやシェル、ファイルなど複数リソースも対象にできる。	Libertyだけでは解決できないので、スクリプトの作成もしくは自動化ツールの併用を検討する。

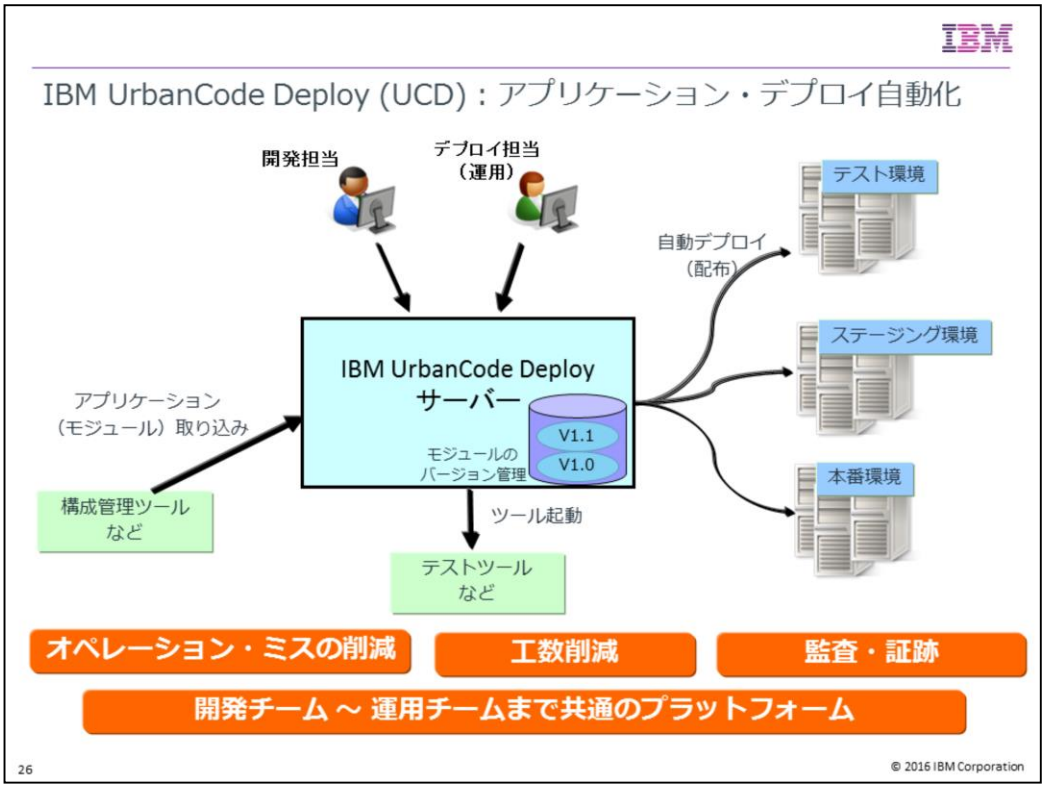
25

「デプロイするターゲット環境」は、スタンドアロン構成、シンプル・クラスター構成、Libertyクラスター構成、またLibertyだけでなく複数ミドルウェアへのデプロイも管理したいという4通りに分けられます。Libertyのトポロジーについては、「トポロジー設計」のセッションを参照ください。

スタンドアロン構成は様々なデプロイ方法を取ることが可能です。詳細は前のセッションの「運用設計」-「アプリケーションのデプロイ」を参照ください。

複数のLibertyサーバー、複数ミドルウェアの複雑なトポロジーに対して一元管理したい場合はLibertyや自動化ツールの機能の適用を検討ください。Libertyクラスター構成ではLiberty Collectiveが配下のメンバーやクラスターを知っているので、FileTransfer MbeanなどLibertyの機能を利用したデプロイが可能です。

一方、シンプル・クラスター構成や複数ミドルウェアへのデプロイを管理したい場合は、Libertyだけでは解決できないので、スクリプトの作成もしくは自動化ツールを検討してください。



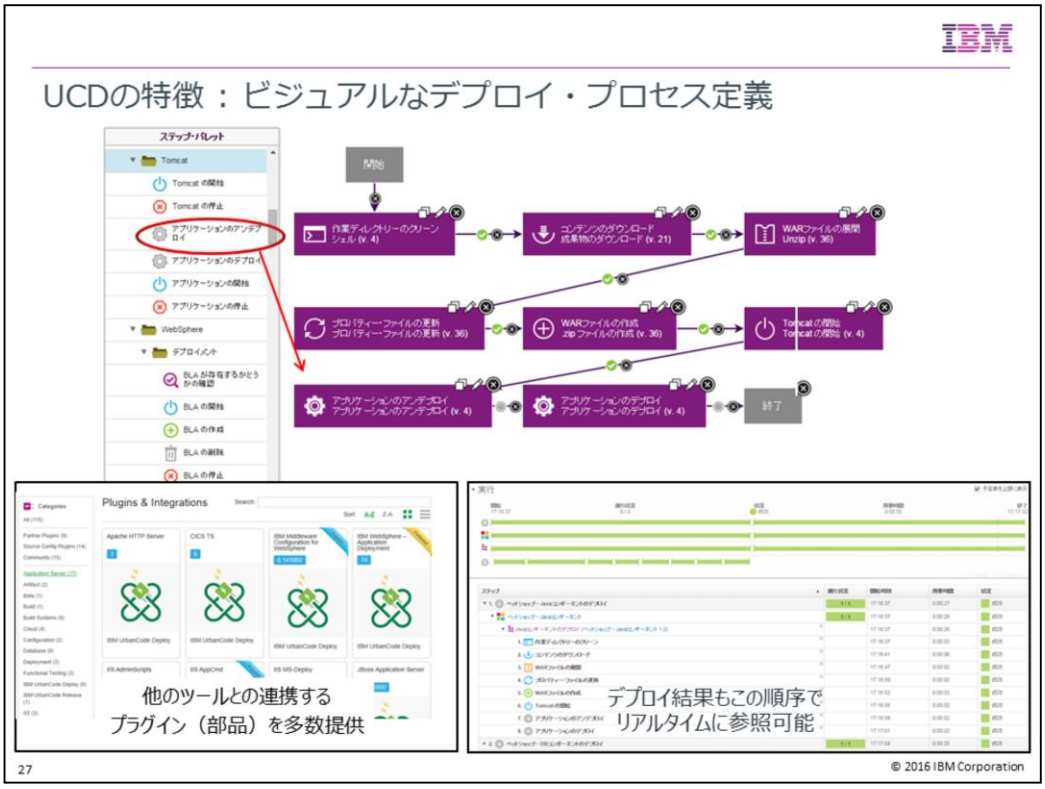
ここからは「アプリケーション・デプロイ自動化」ツールであるIBM UrbanCode Deploy(UCD)を紹介します。

UCDではビジュアルにデプロイ業務を設計することができ、UCDが提供するプラグイン（部品）を活用することで、構成管理ツール、ビルドツール、テスト自動化ツール、アプリケーションサーバー、データベースなどと容易に連携することができます。アプリケーション開発を行い、必要な様々なモジュールと合わせてビルドし、テスト環境にリリースしてテスト自動化ツールと連携してテストを実施、問題がなければ本番環境にリリースする。これらの一連の開発サイクル(DevOpsライフサイクル）をコントロールできるのがUCDです。開発チーム ～ 運用チームまで共通のプラットフォームとして利用できます。通常アプリーケーション・デプロイは手作業で行われることが多いですが、UCDを使うことでミスを削減することも可能です。

UCD適用の効果としては以下の3つがあげられます。

・オペレーション・ミスの削減
・工数削減（スピードアップ）
・監査・証跡（セキュリティー）

developerWorksのUrbanCodeカテゴリーに製品情報があります。
https://www.ibm.com/developerworks/jp/websphere/category/ucd/



UCDの特徴の1つは「ビジュアルなプロセス定義」です。

スーパー・プログラマーが優れたスクリプトを記述しリリース管理として利用しているプロジェクトは多いと思います。これでうまくいっていればよいのですが、その人が抜けた後に引き継いだ担当者が記述内容、つまりリリース・プロセスを理解できず、ブラックスボックス化して保守ができないという厳しい現状を抱えているプロジェクトを聞くこともあります。

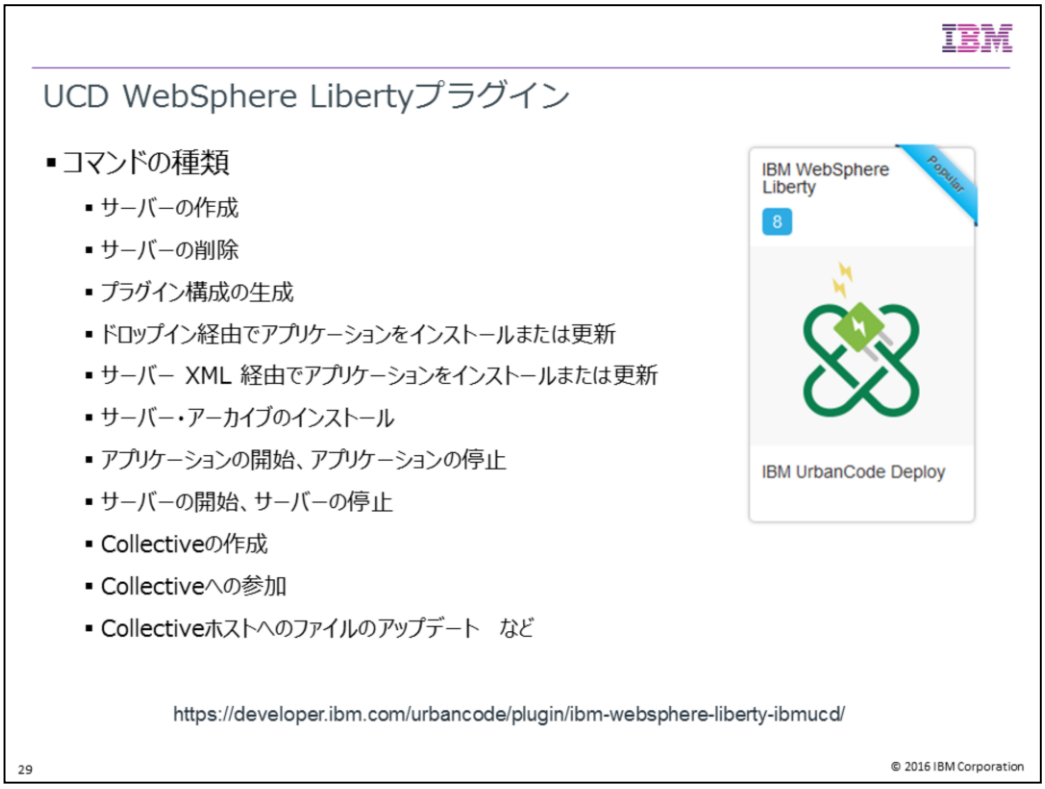
これに対してUCDは、グラフィカルなドラッグ＆ドロップ操作でプロセス定義が可能です。デプロイ結果もこの定義の順序でリアルタイムに参照可能です。

また、アプリケーション・サーバー、データベース、テストツール、シェルなどを操作する多様なプラグイン（部品）が提供されており、すべてを自作する必要がありません。これによりリリース・プロセスが可視化され、開発と運用がお互いに協力しあえる環境の整備につながります。また、運用担当者にとっては実行は自動化しコンピューターに任せて、自身は適切な保守性のよいプロセスは何か、どのように運用管理をするかといった、より重要な設計部分に時間を割けるようになります。

UCDのプラグインの一例を紹介します。多くのデプロイ先のプラットフォームや、開発ツールとの連携が強みであり、複数ミドルウェアへのデプロイを管理できます。

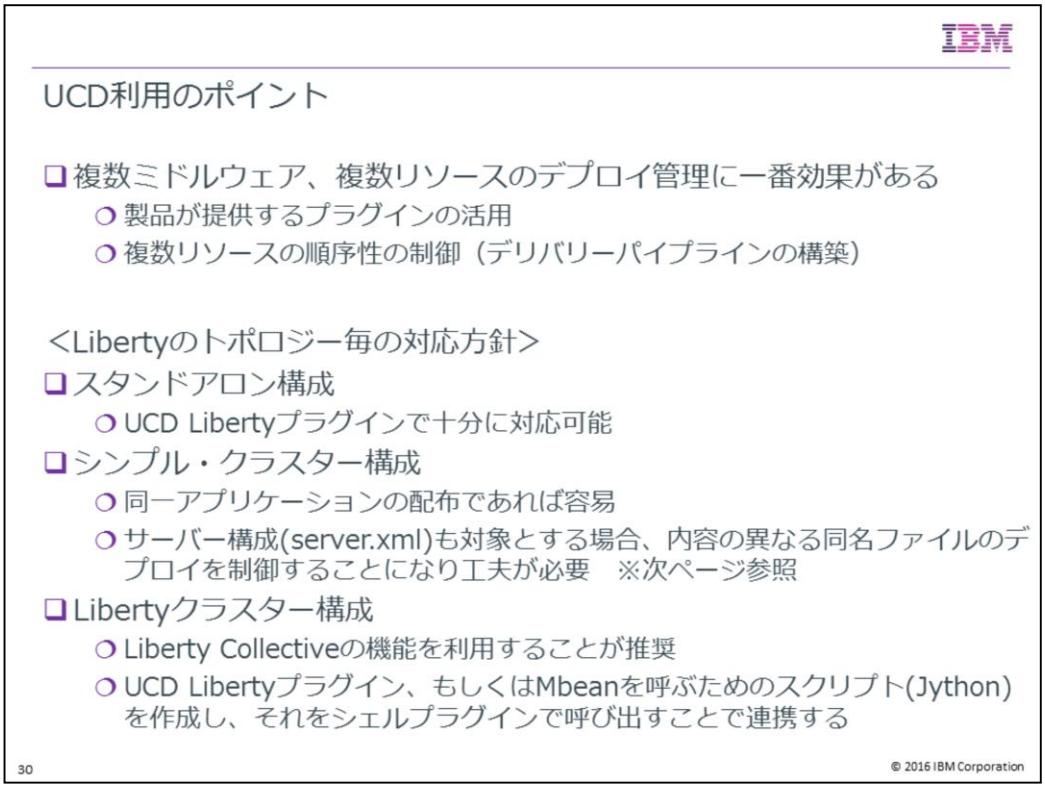
プラグインは2016年8月時点で175種類です。以下のサイトからダウンロード可能です。プラグインは増えておりバージョンも上がりますので、適宜このサイトを確認するようにしてください。

https://developer.ibm.com/urbancode/plugins/ibm-urbancode-deploy/



UCDが提供するLibertyプラグインは以下のサイトからダウンロード可能です。
2016年に入りLiberty Collectiveに対する操作（ステップ）が特に強化されています。

ttps://developer.ibm.com/urbancode/plugin/ibm-websphere-liberty-ibmucd/



UCDとLiberty連携では、複数ミドルウェア、複数リソースのデプロイ管理に一番効果があります。プロジェクトではLibertyだけでなく、IHSやデータベース、シェルなども含めたデプロイを管理するケースも多く、複数リソースの順序性の制御（デリバリーパイプラインの構築）をUCDを活用することにより行うことができます。

ここでUCDを使用する場合のLibertyのトポロジー毎の対応方針をまとめます。

スタンドアロン構成に対しては、UCDのLibertyプラグインで十分対応が可能です。

シンプル・クラスター構成の場合、デプロイ対象をアプリケーションのみにする場合は容易です。ただしサーバー構成(server.xml)も対象とする場合、内容の異なる同名ファイルのデプロイを制御することになり次のページのような工夫が必要です。

Libertyクラスター構成の場合、Liberty Collectiveの機能を利用することが推奨です。UCD Libertyプラグイン、もしくはMbeanを呼ぶためのスクリプト(Jython)を作成し、それをシェルプラグインで呼び出すことで連携します。

参考情報を以下にまとめます。

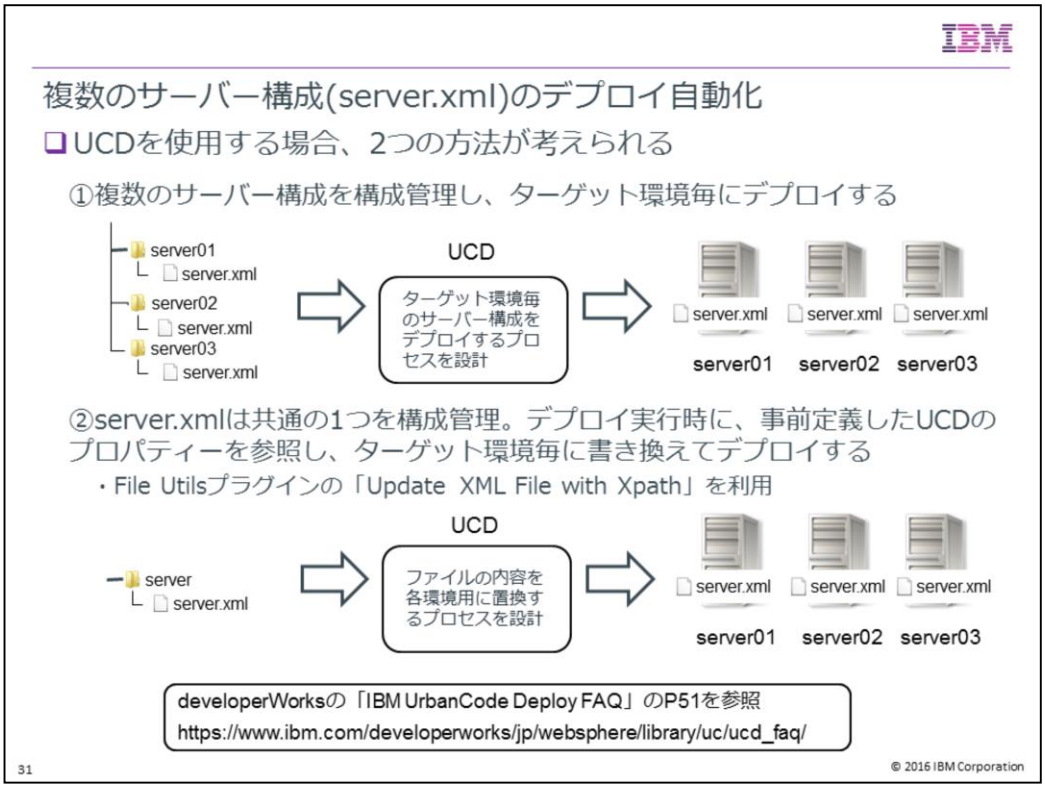
・Deploying Liberty using IBM UrbanCode Deploy (Part 1)

https://developer.ibm.com/wasdev/docs/deploying-liberty-using-ibm-urbancode-deploy-part-1/

・Deploying Liberty using IBM UrbanCode Deploy (Part 2)

https://developer.ibm.com/wasdev/docs/deploying-liberty-using-ibm-urbancode-deploy-part-2/

・UrbanCode Deploy: Collectives and clustering with Liberty
https://developer.ibm.com/wasdev/docs/urbancode-deploy-collectives-clustering-liberty/



シンプル・クラスター構成の場合、デプロイ対象をアプリケーションのみにする場合は容易です。ただしサーバー構成(server.xml)も対象とする場合、内容の異なる同名ファイルのデプロイを制御することになり工夫が必要です。ここで紹介する方法はUCDに限りませんので、他の自動化ツールを使用する場合も参考にしてください。

UCDを使用する場合、2つの方法が考えられます。

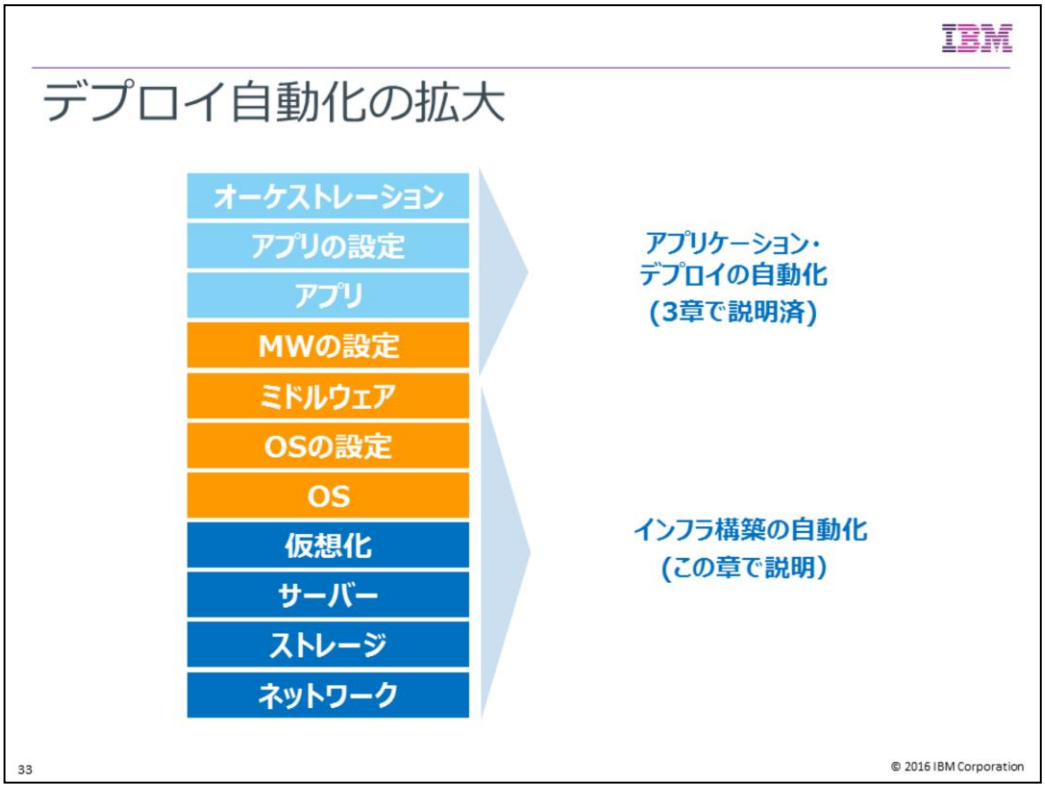
①複数のサーバー構成を構成管理し、ターゲット環境毎にデプロイする

②server.xmlは共通の1つを構成管理。デプロイ実行時に、事前定義したUCDのプロパティーを参照し、ターゲット環境毎に書き換えてデプロイする

詳細はdeveloperWorksの「IBM UrbanCode Deploy FAQ」のP51を参照してください。

https://www.ibm.com/developerworks/jp/websphere/library/uc/ucd_faq/

4. インフラ構築の自動化

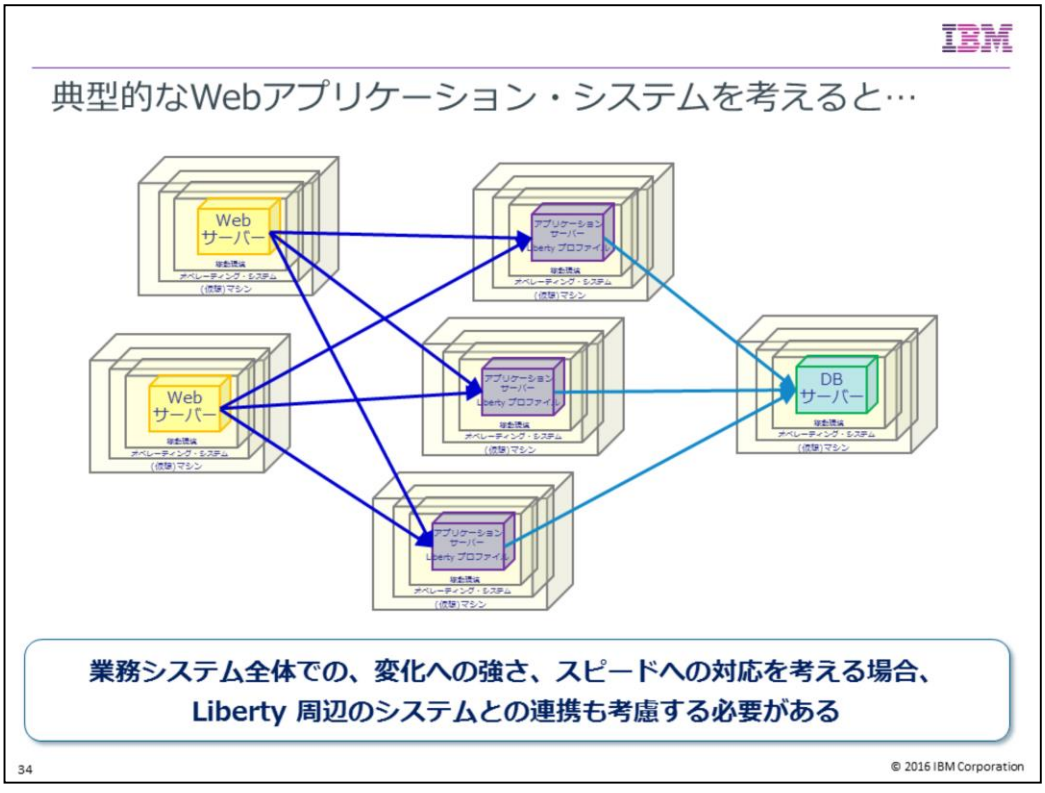


デプロイ自動化を考えた場合、大きく「アプリケーション・デプロイの自動化」と「インフラ構築の自動化」の２つに分けられます。3章ではアプリケーション・ デプロイの自動化を説明しました。

この章では「インフラ構築の自動化」を紹介します。ここには

・ミドルウェア
・OSの設定
・OS
・仮想化
・サーバー
・ストレージ
・ネットワーク

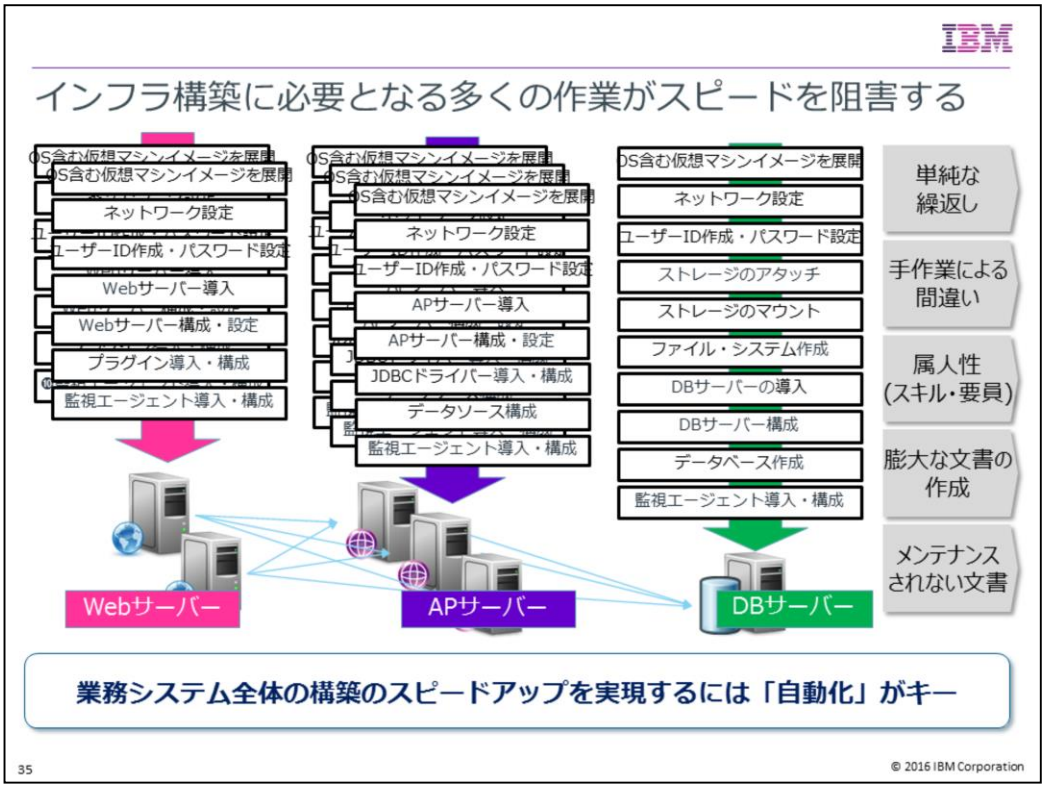
が含まれます。



ここで、ある業務システムとして、典型的なウェブ・アプリケーション・システムの構築を考えてみます。

ある一つのサービスを提供するために、複数のサーバーで複数のコンポーネントが稼動し、それらが連携した構成になることは一般的です。

したがって、Liberty が稼動するサーバーだけでなく、その周辺にあるコンポーネントやツールを含めた業務システム全体で、変化やスピードに対応できるようにしなければなりません。



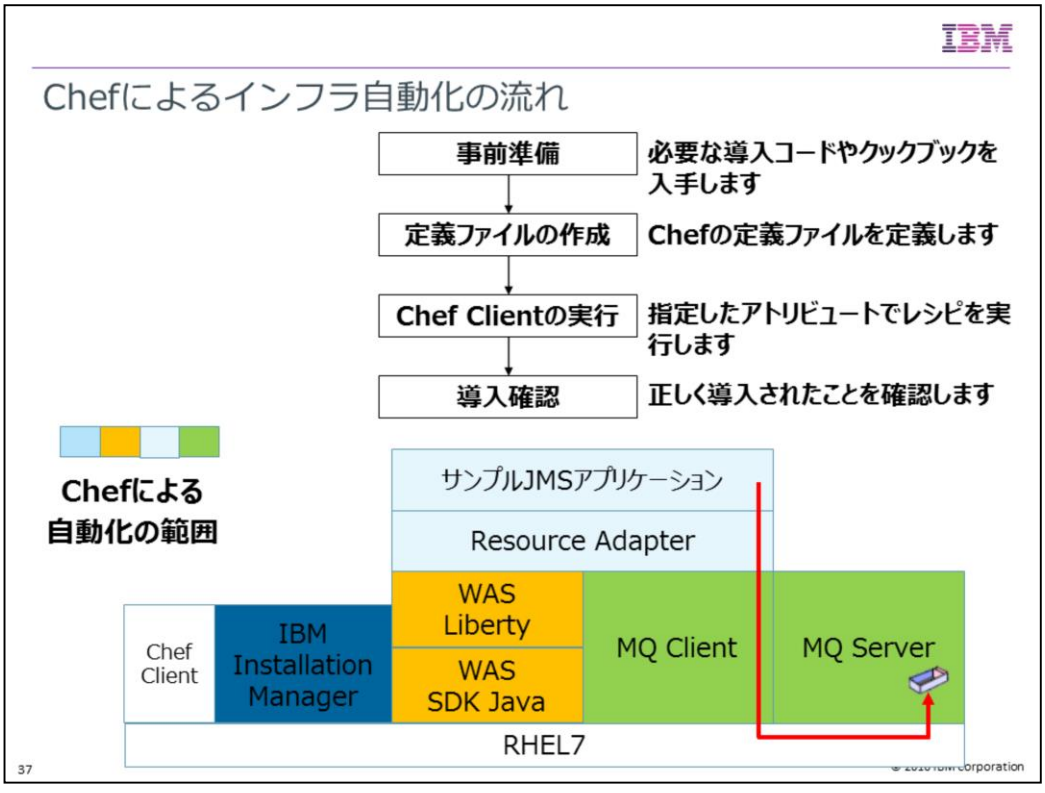
しかし、多くのシステム開発プロジェクトではシステム構築には数週間から数ヶ月かかる場合もあります。

いくら Liberty 自身を簡単に構築できたとしても、業務システム全体の構築に数ヶ月もかかっていては、変化の激しいビジネス環境に対応することはできません。

システム構築が長期化する一番大きな理由は、システム構築に必要な多くの作業が手作業で行われていることです。しかも、単純な作業の繰返しが作業負荷を増やし、手作業によるミスが発生しやすいという問題もあります。また、それぞれのシステムの構築にはそのミドルウェアのスキルを持った担当者が必要であり、要員の調整に時間がかかったり、スキルレベルの違いによる作業品質のばらつきが発生したりします。こういった品質の問題を解決するために、膨大なドキュメントを作成する時間や工数も問題です。さらに、システムの変更のたびに、これらのドキュメントを確実にメンテナンスしていくことは非常に困難です。

これらの問題を解決するのが、システム構築の「自動化」です。自動化は、システム構築の品質を上げながら、システム構築作業を効率化し、Liberty を含めたシステム全体の構築をスピードアップすることができます。

インフラ構築の自動化の代表的なテクノロジーとしてOSSのChef, Docker、IBM製品のPureApplicationの3つ紹介します。ChefやDockerはOSが用意された環境に対してインフラのデプロイ自動化を行います。一方、PureApplicationはその下の仮想化、ストレージ、ネットワークのレイヤーの構築自動化も可能です。各々の特徴をまとめました。



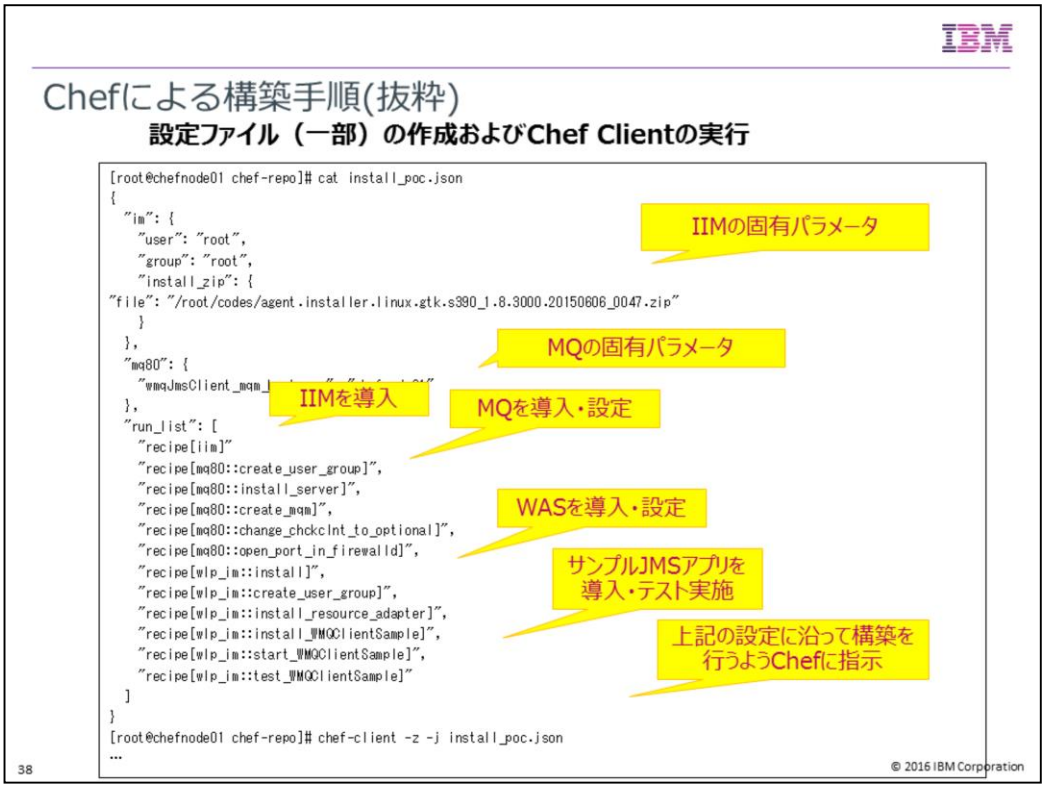
Chefはクックブックというファイルの集まりで管理します。Chefによるインフラ自動化の流れはこのようになります。

①事前準備

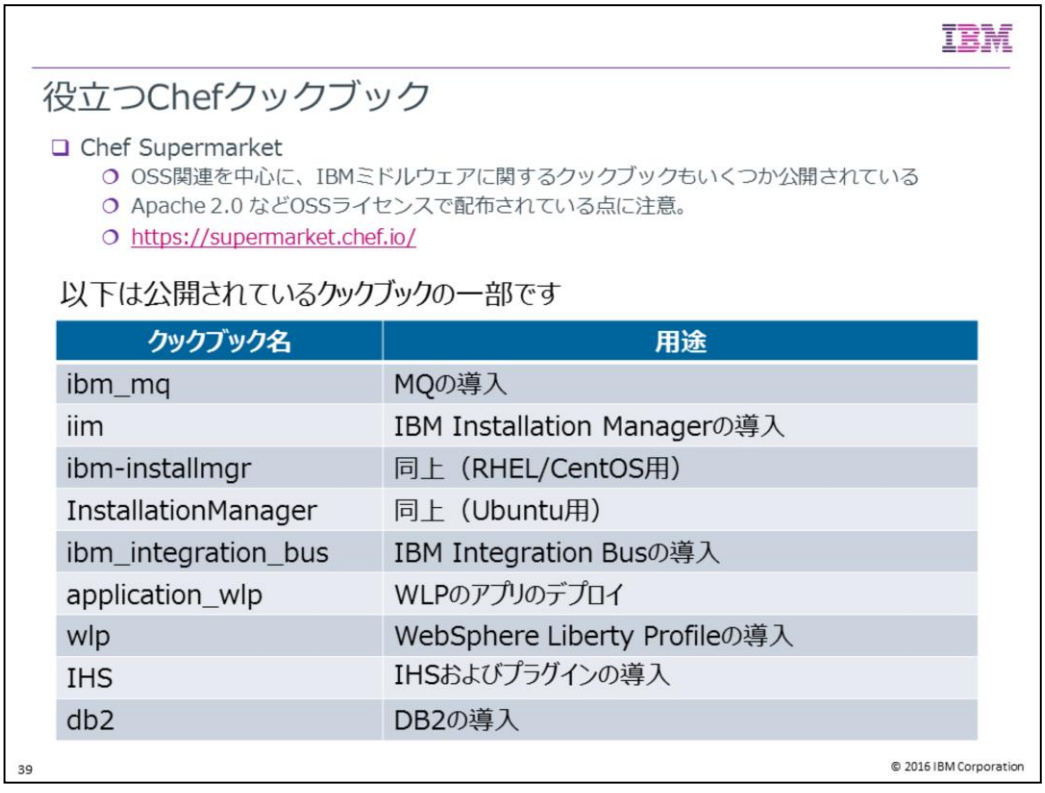
②定義ファイル作成

③Chef Clientの実行

④導入確認



Chefの設定ファイル（一部）の作成およびChef Clientの実行のイメージです。

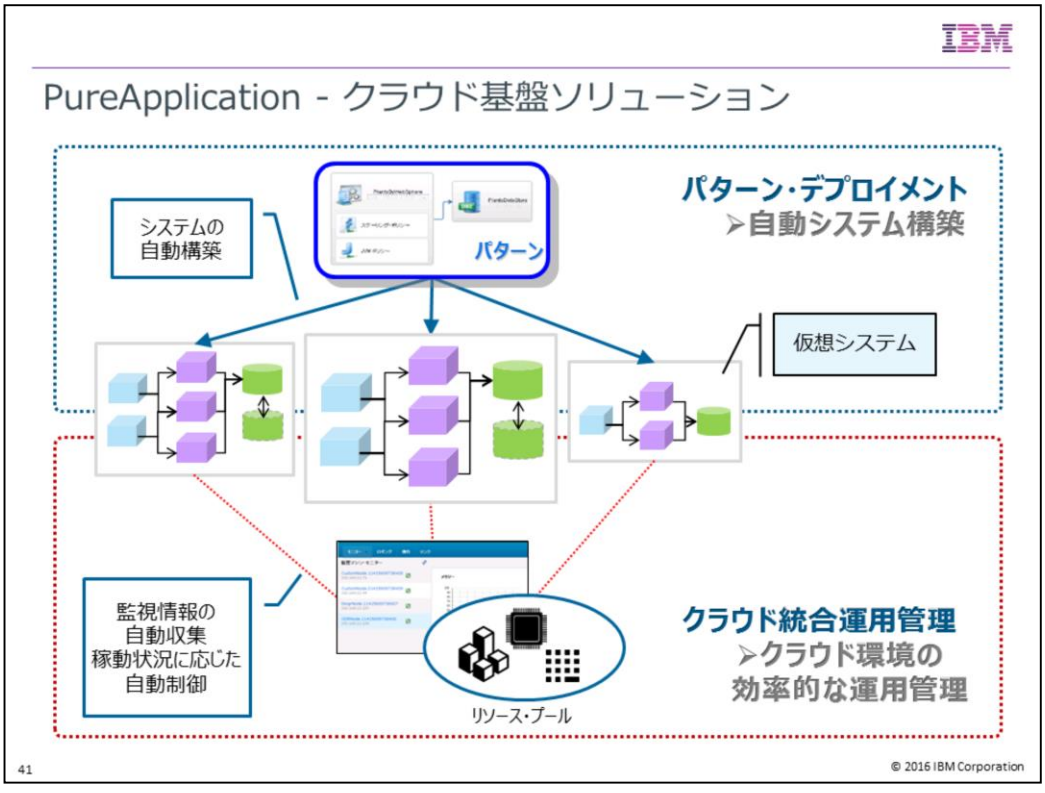


Chefには様々なミドルウェアの導入・設定を行うクックブックが存在します。
例えば、
IBMミドルウェア　WAS、MQ、IBM Integration Bus(WMB) など
それ以外のミドルウェア　Oracle、MySQL、PostgreSQL apache2、zabbixなど
OSのカスタマイズに利用する操作もほぼ網羅されています。

新しい技術を展開する手順も続々とクックブック化されています。

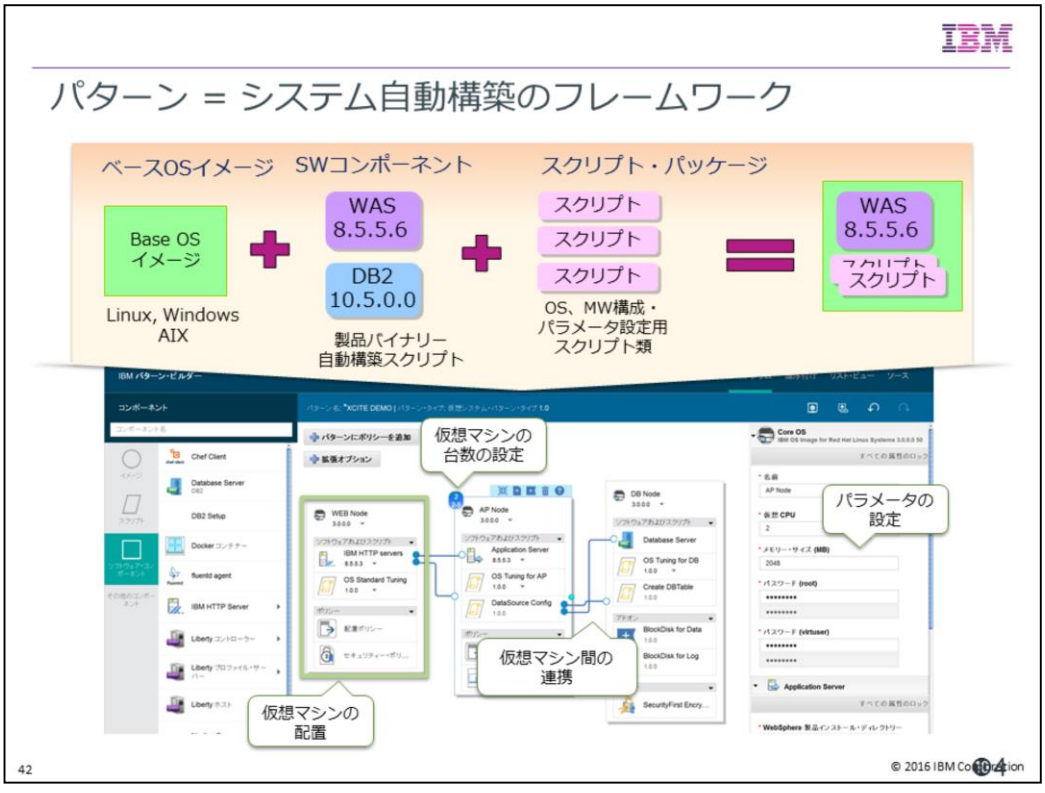
LibertyのDockerイメージをDockerHubに公開しています。このイメージを利用すると、テスト済みのLibertyイメージを開発だけでなく、本番にも利用可能です。

https://hub.docker.com/_/websphere-liberty/



システムの構築の自動化やシステムの運用管理の効率化を実現するのが PureApplication です。PureApplication は、あらかじめ確保されたリソース・プールから必要なリソースを割り当てながら、業務システム全体の構築を自動化します。

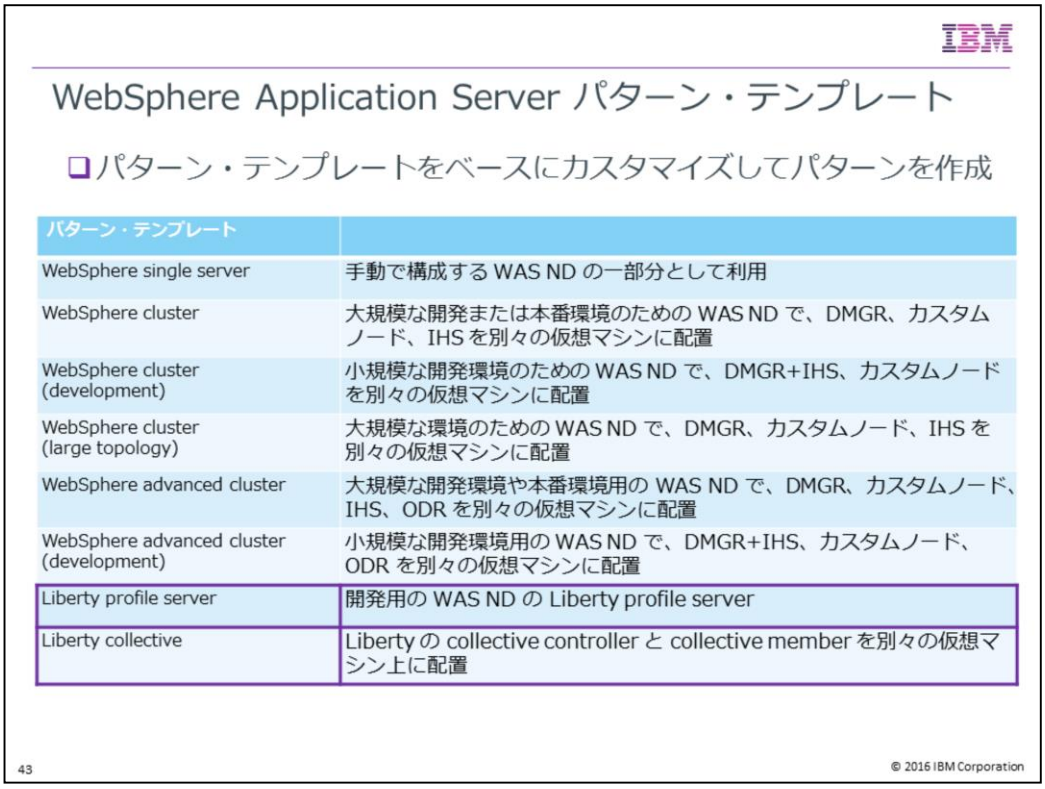
また、システム構築の自動化を実現しているのが、「パターン・デプロイメント」技術です。PureApplication には、デプロイ後の仮想システムを管理するために必要な一連の運用管理機能も備えており、稼動状況のモニタリングし、その状況に応じた自律的な運用を行うこともできるようになっています。



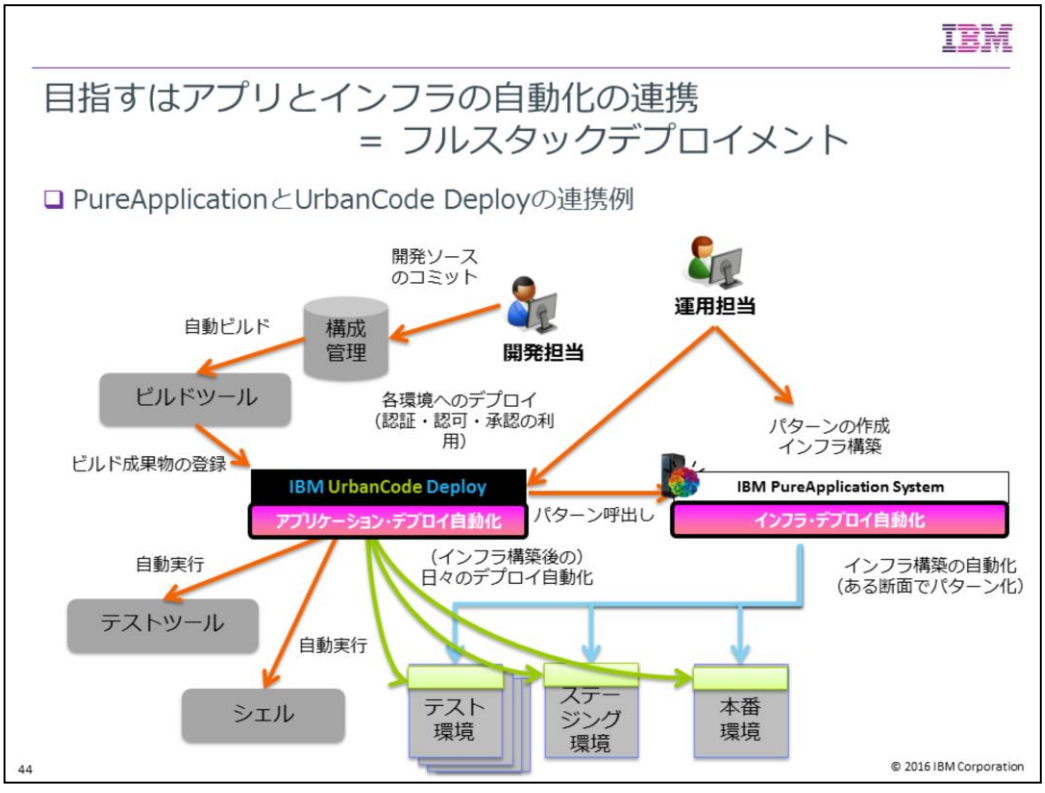
ここで、もうすこし具体的な「パターン」の中をみてみます。

図のように、OS を含んだ仮想マシンの上に、ソフトウェア・コンポーネントを配置します。このソフトウェア・コンポーネントは、WAS や DB2 をはじめとしたミドルウェアです。パラメータ設定やテーブルの作成などの作業手順はスクリプト・パッケージとして部品化しておいたものを利用します。アプリケーション・サーバーからデータベース・サーバーへ接続するための、データソースの定義も同様です。

Liberty は、ソフトウェア・コンポーネントとして、Liberty Core, Liberty Profile Server, Liberty Collective の Controller や Member などを利用可能です。



また、パターン・テンプレートとして、典型的なパターンのサンプルが提供されています。実際には、これらを参考にして、必要最低限のカスタマイズを行ってパターンを作成していくことになります。



ここまでデプロイ自動化として「アプリケーション・デプロイの自動化」と「インフラ構築の自動化」の２つを見てきました。目指すはアプリとインフラの自動化の連携＝フルスタックデプロイメントによる「Immutable Infrastructure」を実現です。PureApplicationとUrbanCode Deployの連携のイメージを紹介します。

Product Demo: PureApplication and UrbanCode Deploy

https://developer.ibm.com/urbancode/videos/product-demo-pureapplication-and-urbancode-deploy/

このセッションの全体のまとめです。

WebSphere Application Server Liberty