IBM

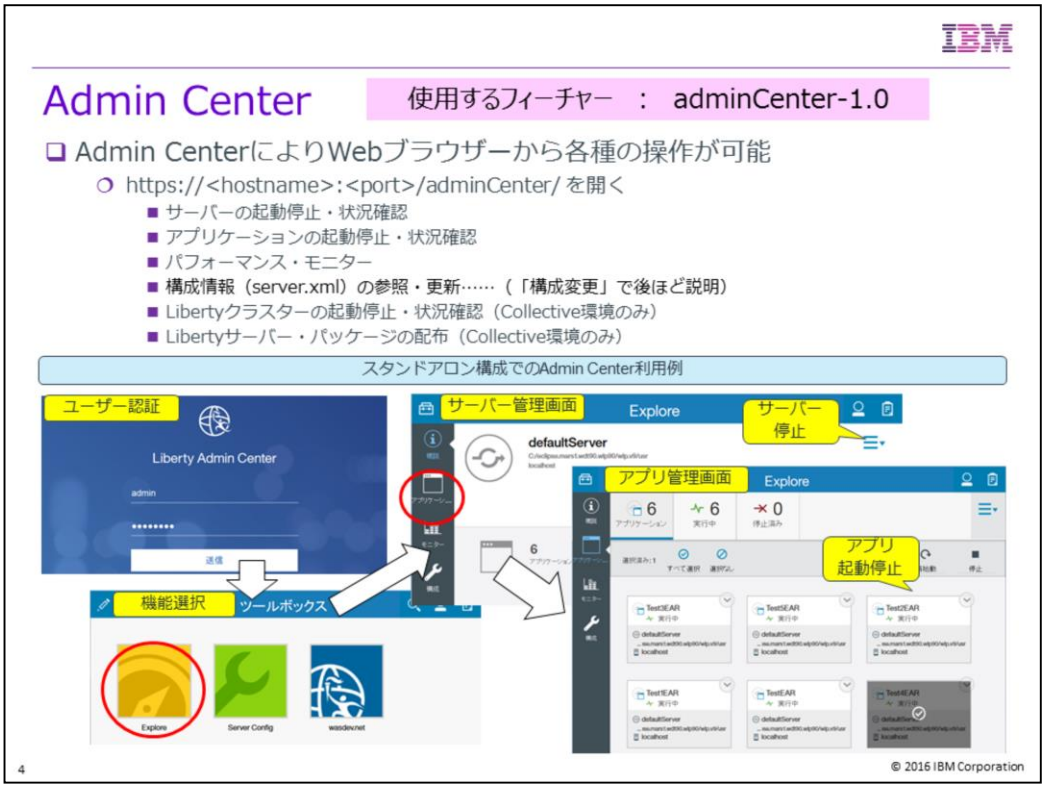
WebSphere Application Server V9

Liberty 基盤設計セミナー

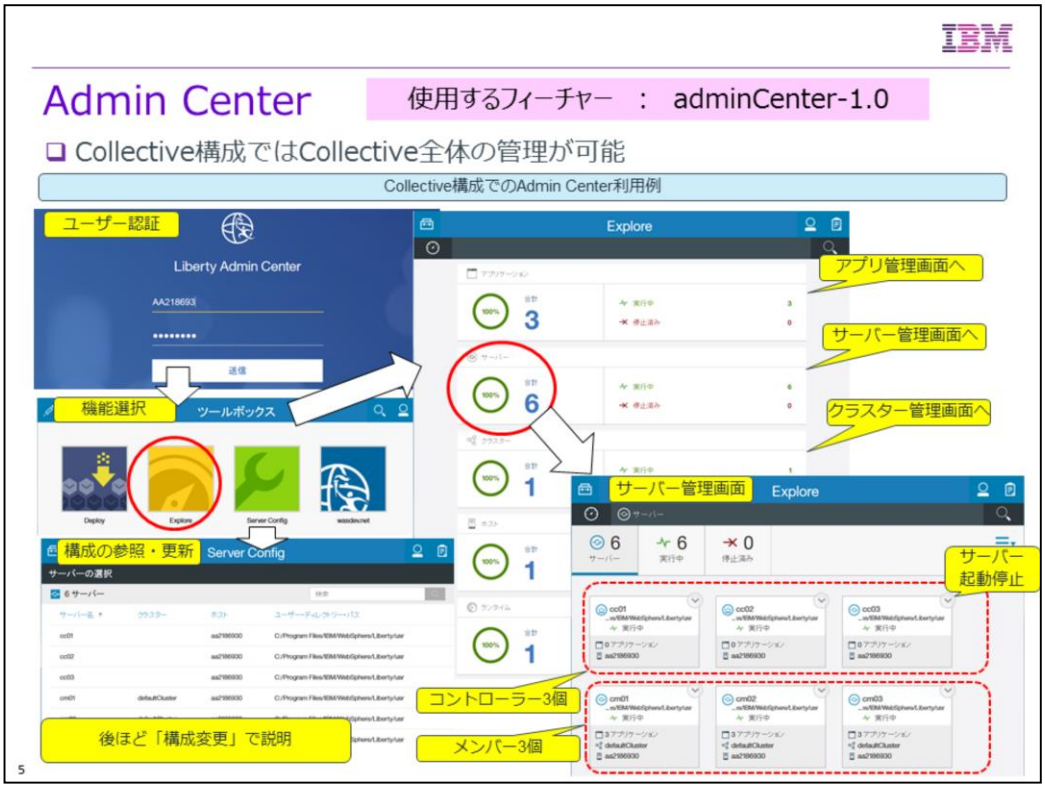
運用設計

アジェンダです。ログ・ログ監視については、「問題判別」のセッションを参照してください。

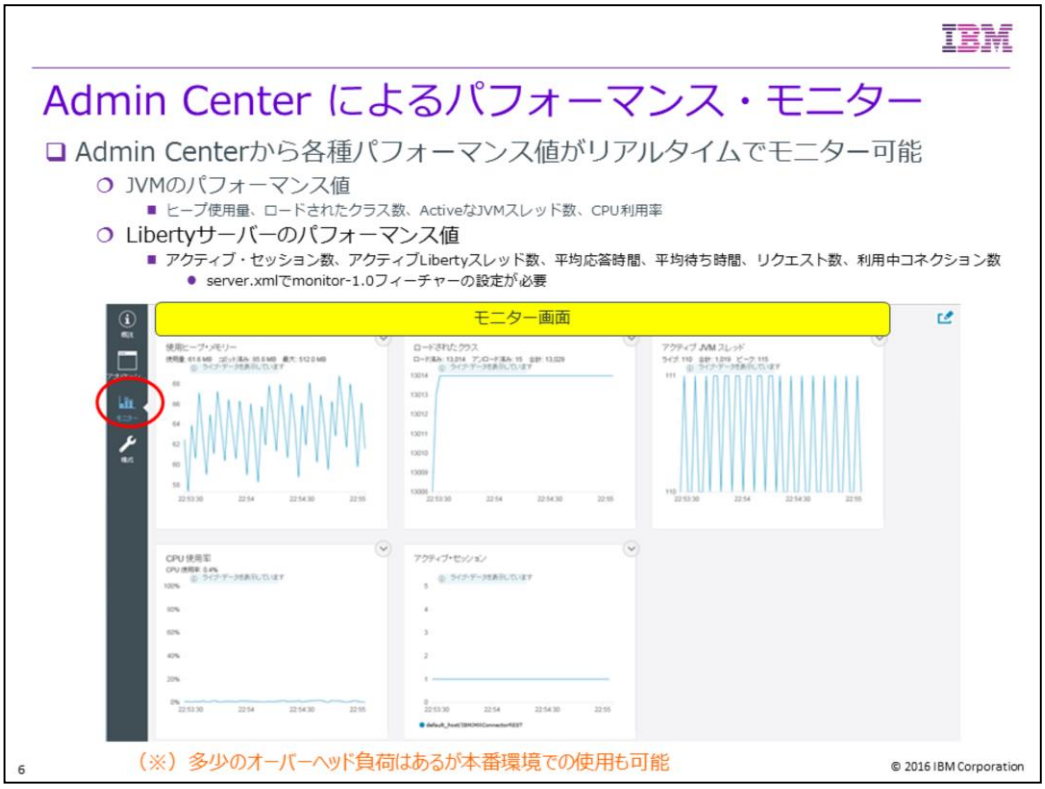
Admin Center



Admin CenterはWAS traditionalの管理コンソールに類似した位置づけの機能で、Webブラウザーから各種の操作が可能になっています。Admin Centerが登場した当初は限られた機能でしたが、継続デリバリーにより機能が拡充されてきました。アクセスURLや実施可能な項目はチャートの通りです。構成情報の参照・更新機能については、後ほど「構成変更」のセクションで説明します。また、スタンドアロン構成での画面遷移の例を、チャートの中に示します。使用するフィーチャーは、adminCenter-1.0です。

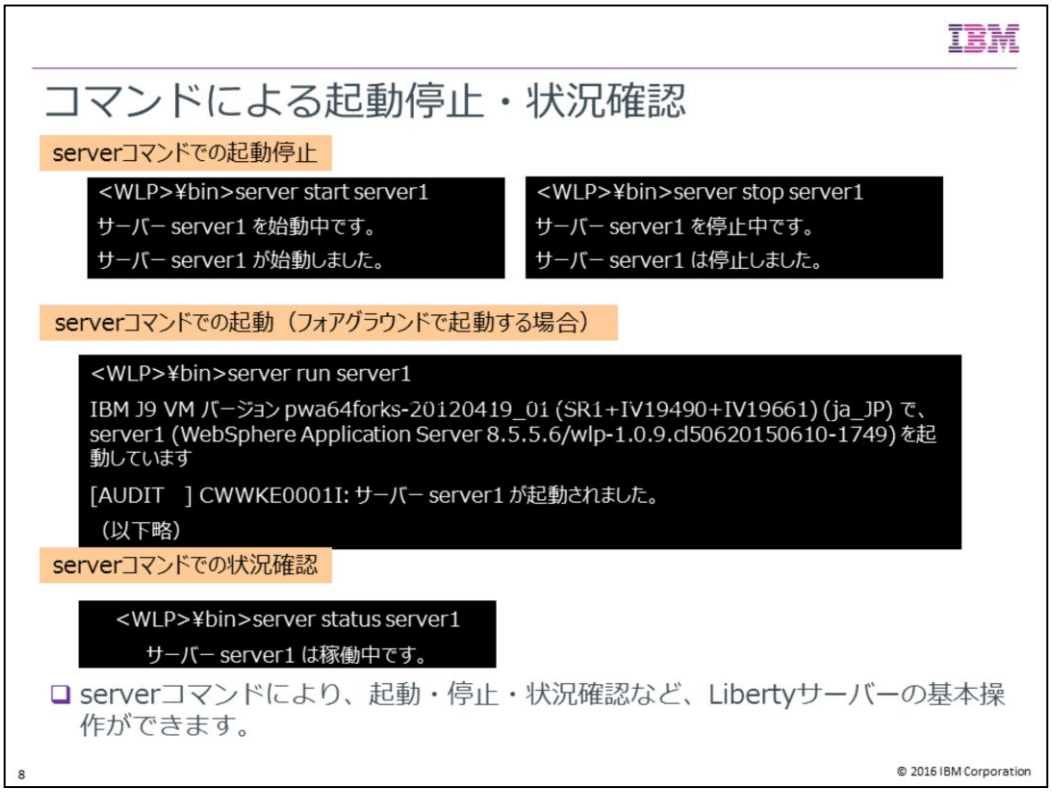


Collective構成でのAdmin Centerにアクセスするブラウザ画面の遷移の例を示します。Explore画面では、アプリケーション、サーバー、クラスター、ホスト、ランタイムがメニューに表示される点がスタンドアロンとは異なります。また、サーバー画面ではこのCollectiveに含まれる全てのコントローラー、全てのメンバーが表示され、操作できることが分かります。

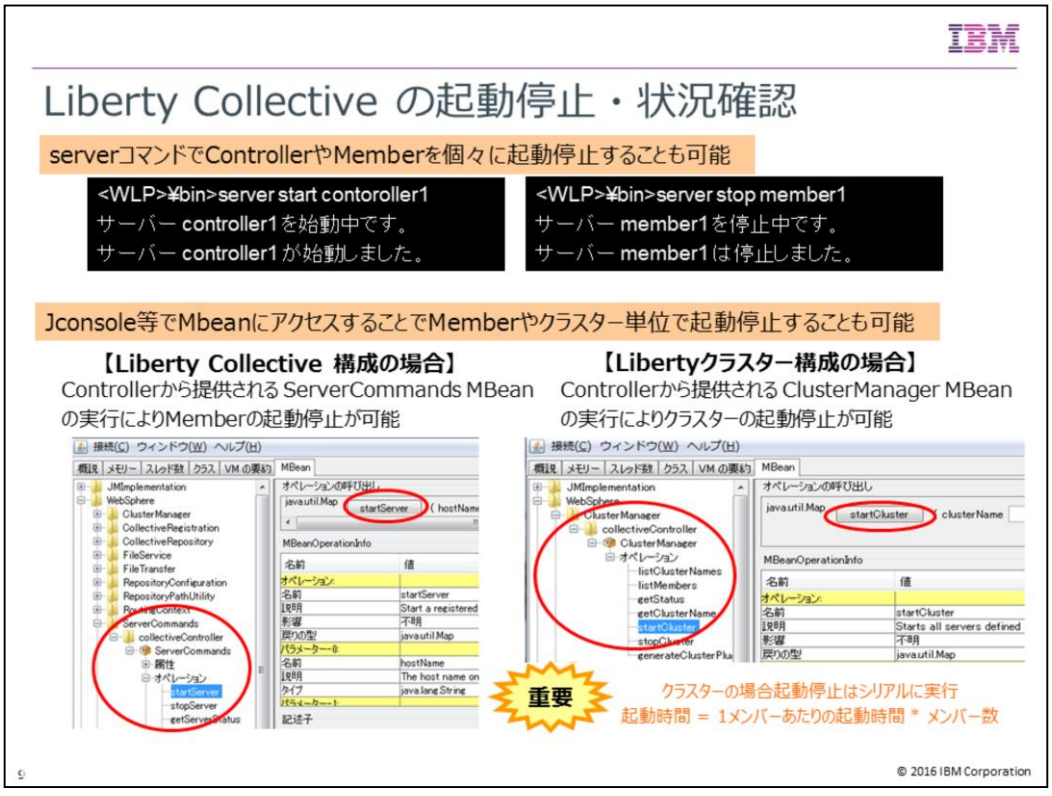


Admin Centerのパフォーマンス・モニター機能では、JVMのパフォーマンス値とLibertyサーバーのパフォーマンス値がリアルタイムでモニターできます。モニター可能な項目はチャート中の通りです。リアルタイム・モニターは多少のオーバーヘッド負荷がありますが、本番環境での使用も可能です。

起動停止・状況確認

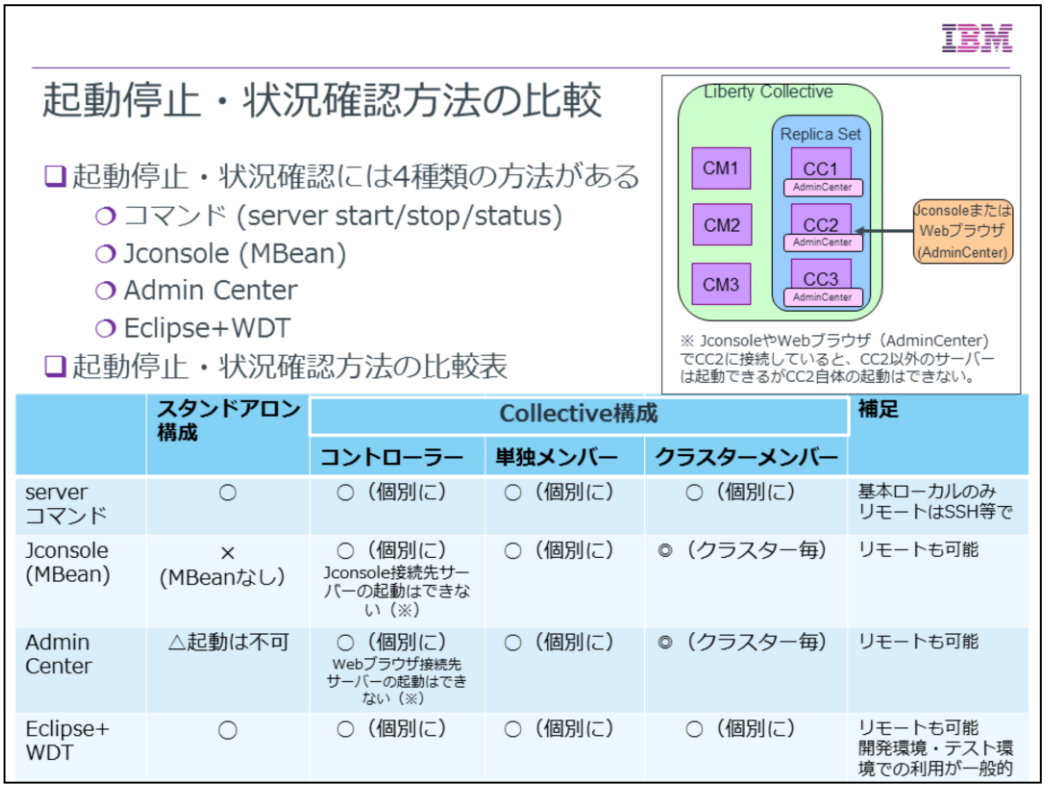


Libertyサーバーは、serverコマンドのstartアクション・stopアクションを実行して起動・停止を行うことができます。起動には、startアクションに加えて、runアクションを利用することもできます。runアクションではフォアグラウンドでLibertyが起動されるため、稼働中はプロンプトは戻りません。また、Libertyサーバーの状況確認には、statusアクションを利用します。



Liberty Collective 構成の場合、Collective Member 個々は前ページと同様にserverコマンドのstartアクション・stopアクションを実行して起動・停止を行うことができるのに加えて、Collective Controller で提供されるServerCommands MBean のstartServer・stopServerのオペレーションを実行することで、Collective Member の起動・停止を行うこともできます。

Libertyクラスター構成の場合、Collective Controller で提供されるClusterManager MBean のsartCluster・stopClusterのオペレーションを実行することで、Libertyクラスター全体の起動・停止を行います。ここで注意が必要なのは、Libertyクラスター構成の場合、起動・停止は各Memberシリアルに実行されるため、Libertyクラスター全体の起動・停止時間は、1メンバーあたりにかかる起動・停止時間とメンバー数の積になり、Member数が多ければそれだけ多くの時間がかかることになります。



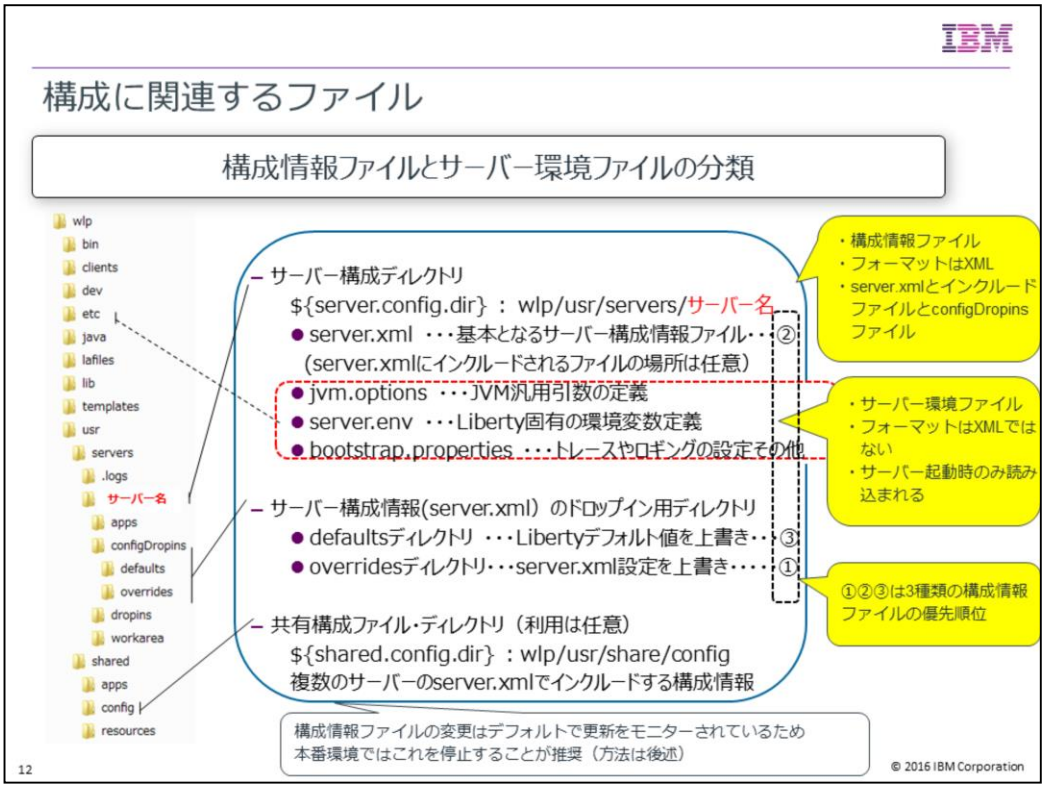
	スタンドアロン構成	Collective構成			補足
		コントローラー	単独メンバー	クラスターメンバー	
server コマンド	○	○（個別に）	○（個別に）	○（個別に）	基本ローカルのみ リモートはSSH等で
Jconsole (MBean)	× (MBeanなし)	○（個別に） Jconsole接続先サーバーの起動はできない（※）	○（個別に）	◎（クラスター毎）	リモートも可能
Admin Center	△起動は不可	○（個別に） Webブラウザ接続先サーバーの起動はできない（※）	○（個別に）	◎（クラスター毎）	リモートも可能
Eclipse+ WDT	○	○（個別に）	○（個別に）	○（個別に）	リモートも可能 開発環境・テスト環境での利用が一般的

起動停止・状況確認についてまとめます。方法としては、チャートの通り4種類あります。

これを比較したのがチャート中の表です。右上の図はCollective構成において、JconsoleやAdmin Centerの接続先であるコントローラーは、停止はできるが起動はできないことを説明しています。共に、それ以外のコントローラーやメンバーは起動・停止ができます。

比較表から分かる通り、クラスターの扱いに関しては、JconsoleやAdmin Centerが優れていますが、全般的に見た場合、コマンドとWDTが優れています。また、WDTについては後ほど「構成変更」でも説明します。

構成変更



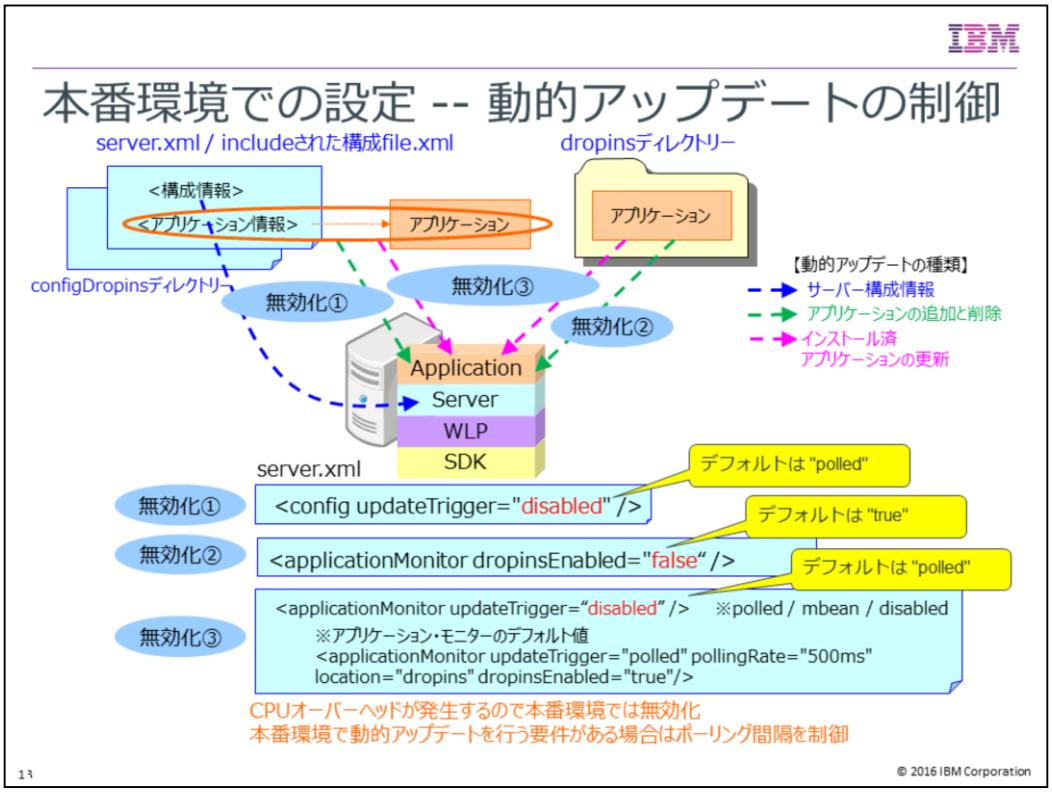
構成に関連するファイルを復習を兼ねて、簡単にまとめておきます。

大きく分けると、構成情報ファイルとサーバー環境ファイル（赤点線枠内）の2種類があります。前者はXMLフォーマットであり、設定により動的更新が可能なファイル郡です。後者はXMLフォーマットではないファイル郡で、サーバー起動時のみ読み込まれる点で、前者とは扱いが異なります。

構成情報ファイルの基本となるファイルはserver.xmlファイルで、これにインクルードされるファイルは任意の場所に置くことが可能です。他のサーバー構成と共有される場合には usr/shared/config の下に置くのが一般的です。

また、configDropinsの下にはserver.xmlを補足する位置づけの構成情報ファイルを置くことができます。configDropins/defaults はLibertyの持つデフォルト値を上書きします。また、configDropins/overrides はserver.xmlを上書きします。

優先順位の観点では、点線枠内の①②③の通りです。



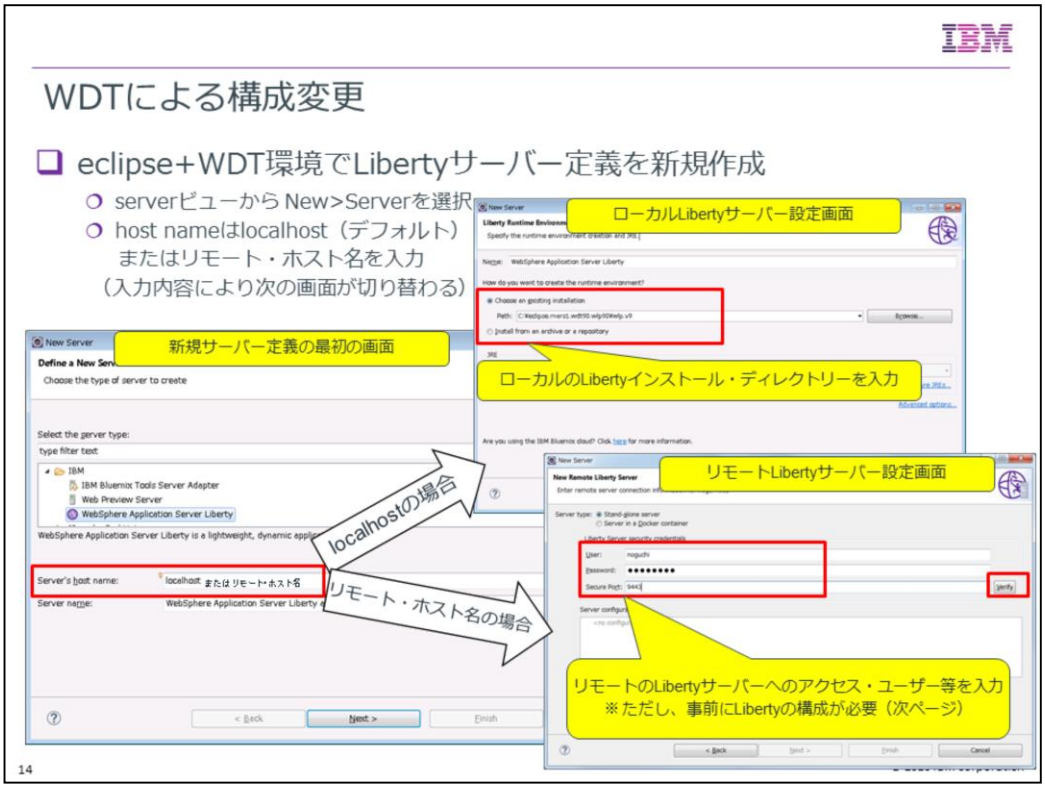
Libertyプロファイルは、サーバー構成やアプリケーション情報を動的にアップデートすることができますが、その機能を必要に応じて制御することができます。

動的アップデートの種類としては、サーバー構成情報、アプリケーションの追加と削除、インストール済アプリケーションの更新、の3つがあります。サーバー構成情報は、server.xmlあるいはincludeされた構成ファイルとconfigDropinsディレクトリーに置かれた構成ファイルから動的アップデートを行うことができます。アプリケーションの追加と削除およびインストール済アプリケーションの更新は、server.xmlあるいはincludeされた構成ファイルやアプリケーション配置ディレクトリー、dropinsディレクトリーから動的アップデートを行うことができます。これらの動的アップデート機能に関して、それぞれの方法で無効化あるいはモニターのポーリング間隔の変更を行うことができます。

構成情報をモニターする構成サービスやアプリケーション・モニターによるポーリングはオーバーヘッドが発生するので、本番環境では無効化します。本番環境で動的アップデートを行う要件がある場合は、ポーリング間隔を長く設定するなどして、できるだけオーバーヘッドが少なくなるように対応します。

【KCリンク】

動的更新の制御:
http://www.ibm.com/support/knowledgecenter/ja/SSAW57_9.0.0/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_dyn_upd.html

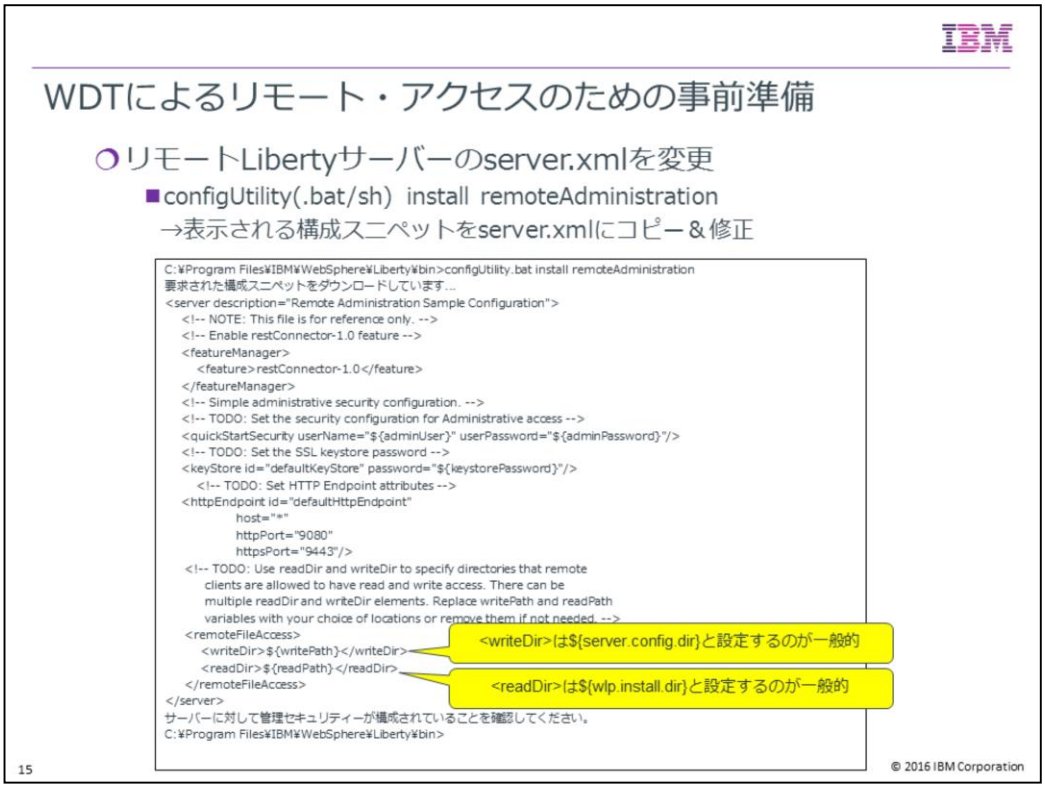


WDTはEclipseのプラグインで、Libertyサーバー用のアプリケーションを開発したり、構成関連ファイルを作成・編集するのに必要なツールです。従来からローカルのLibertyサーバーを操作することはできましが、比較的最近になって、リモートのLibertyサーバーにも対応しました。

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。

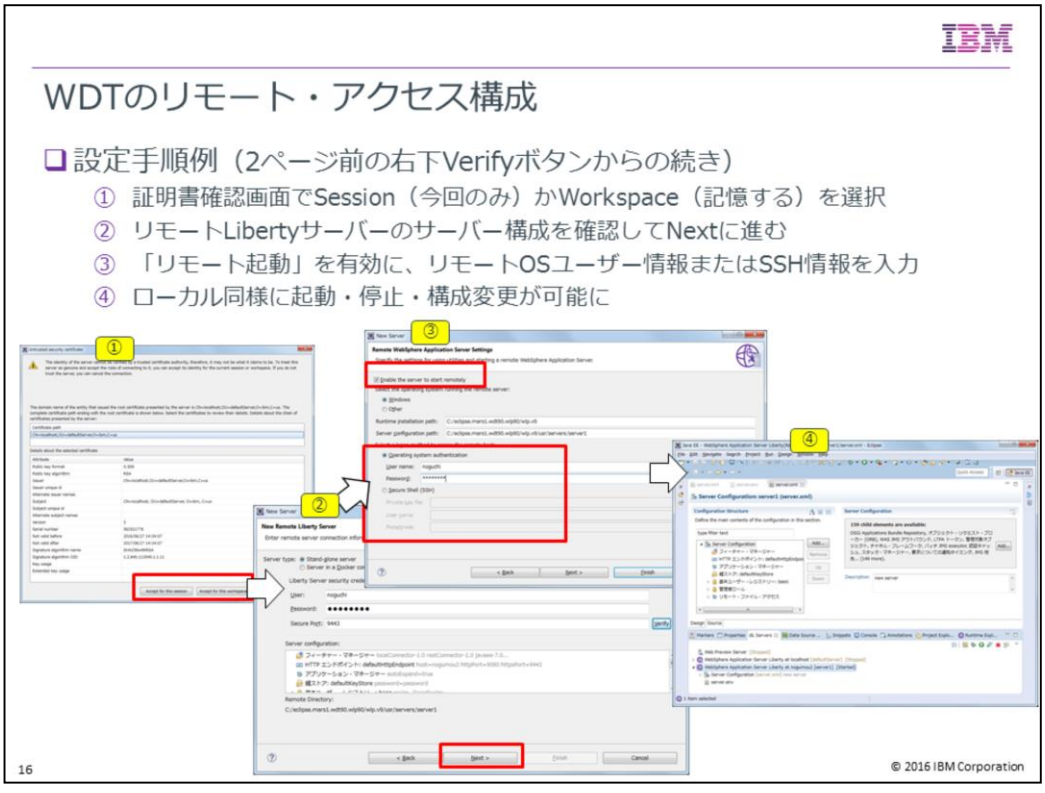
リモートLiberty設定画面では、リモートにあるLibertyサーバーのアクセス情報として、ユーザー名、パスワード、ポート番号を入力して、「Verify」を押します。

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なため、次のページで、その設定方法を説明します。

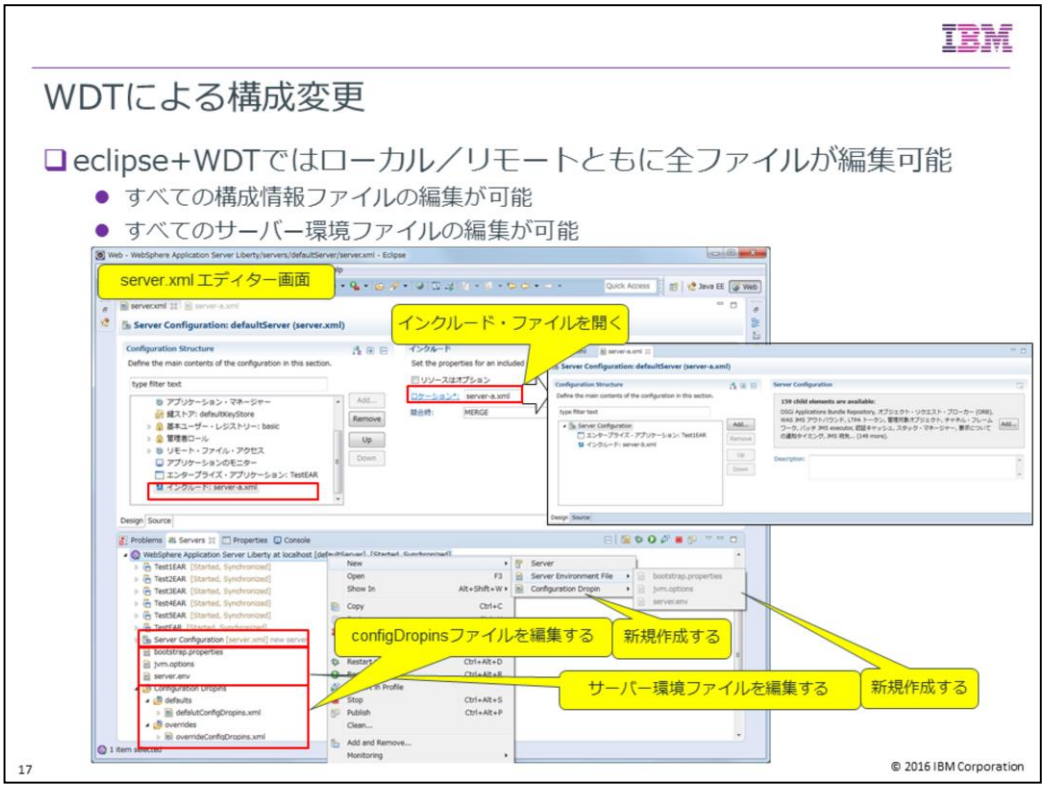


WDTでリモートからアクセスするためには、Libertyサーバーのserver.xmlを更新します。そのための構成スニペットを表示するためには、configUtilityをチャートの通り実行します。また、画面に表示される構成スニペットを枠内に示します。

<remoteFileAccess>要素以下の<writeDir>および<readDir>は黄色噴出しのように設定するとWDTからのアクセスが成功します。



リモートLibertyの設定が完了していると、WDTでのリモートLiberty定義が可能です。チャート中の説明および画面遷移①～④の通りに設定すると、Eclipseのサーバー・ビューにリモートLibertyサーバーを登録できます。

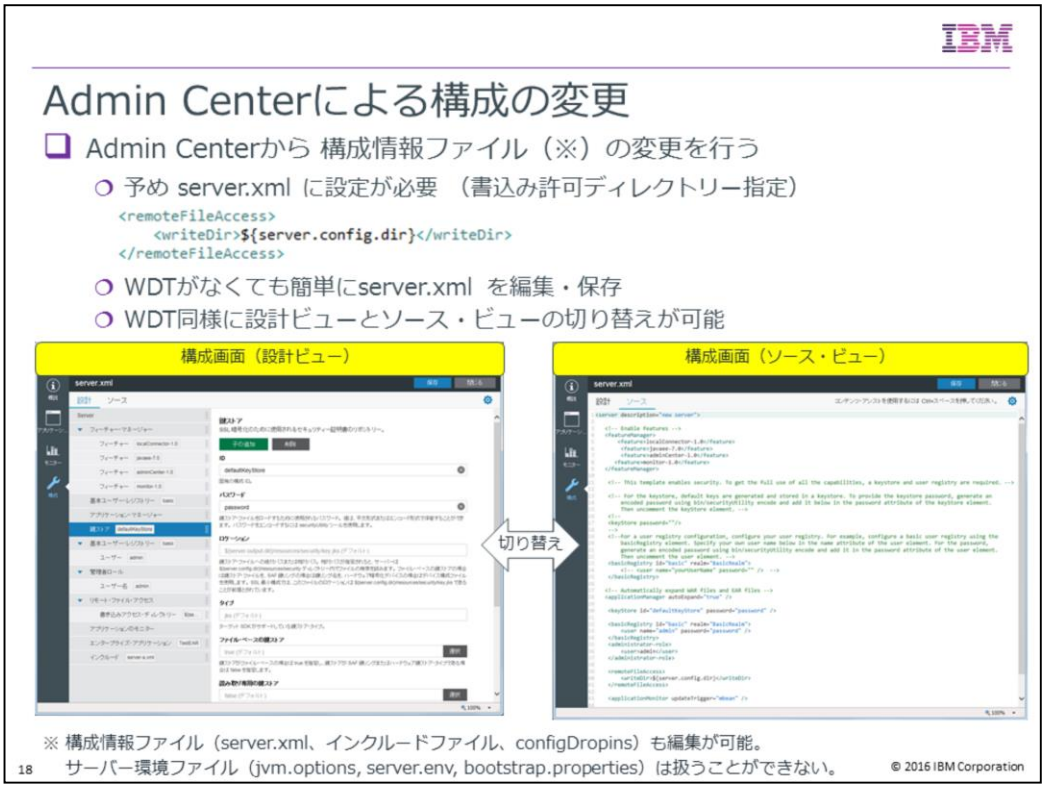


eclipse+WDTを使うとローカル／リモートのLibertyサーバーともに、構成関連ファイルである構成情報ファイルとサーバー環境ファイルの全てを作成・更新することが可能です。

構成情報ファイルのserver.xmlはLibertyサーバーを新規作成したタイミングで作成されていますが、これにインクルードするファイルも、<include>要素を追加することで編集可能になります。インクルード・ファイルはネストさせることも可能です。

configDropinsの構成情報ファイルも、画面キャプチャの通り、サーバー・ビュー内でのマウス右クリックから New > Configuration Dropinsから作成できます。

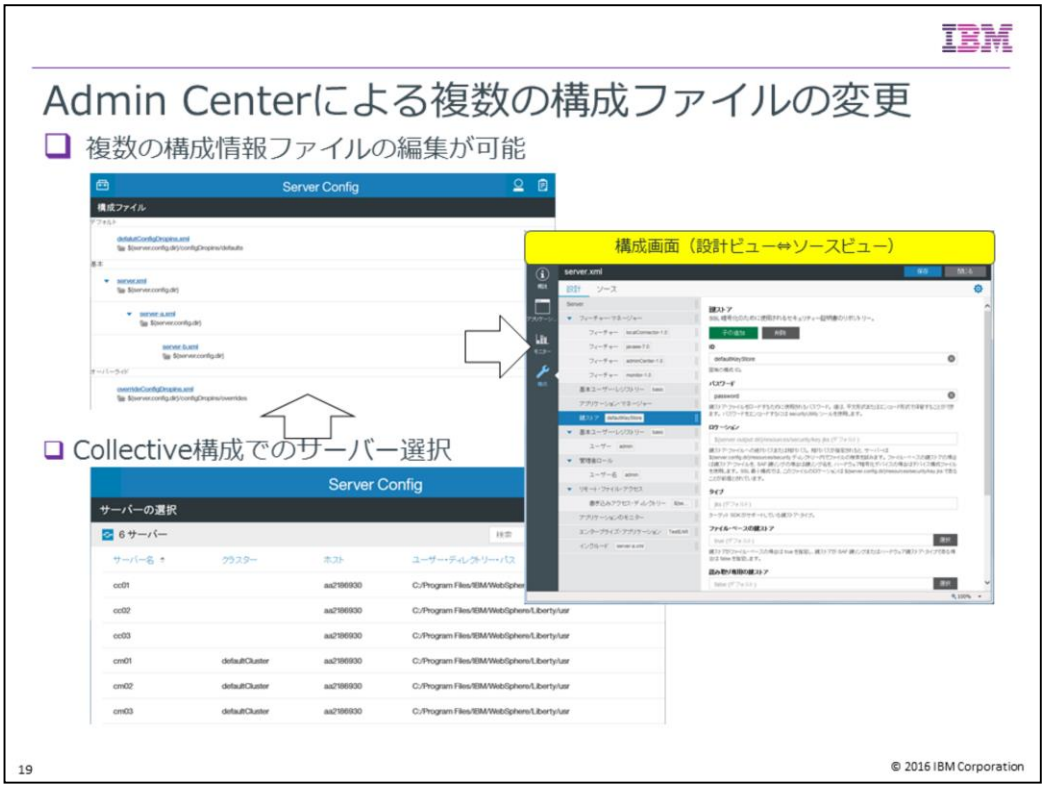
また、サーバー環境ファイルの3種類についても同様に、New > Server Environment File から新規作成することができます。



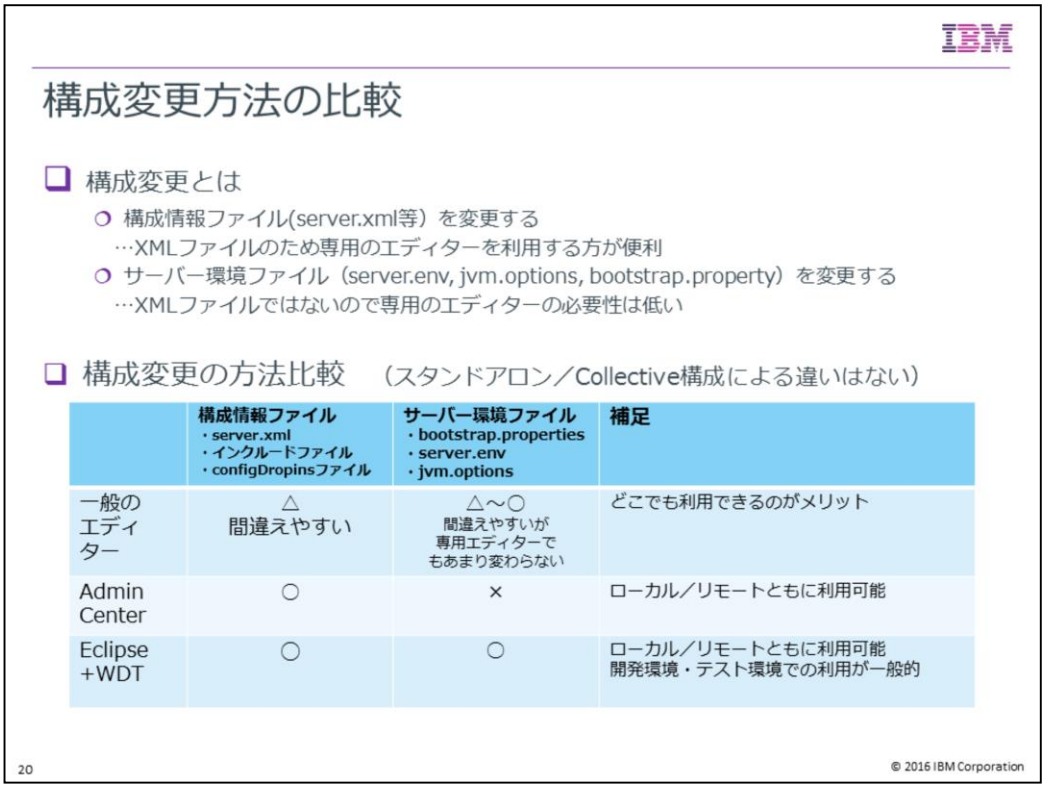
Admin Centerで構成変更を行うためには、チャート中の通り、server.xmlを変更しておきます。

Admin Centerでの構成変更画面でも、WDT同様に、設計ビューとソース・ビューを切り替えて、編集することがきます。

注意点として、Admin Centerでの構成変更では、構成情報ファイル（XML）は扱うことができますが、サーバー環境ファイル（非XML）は扱うことができません。



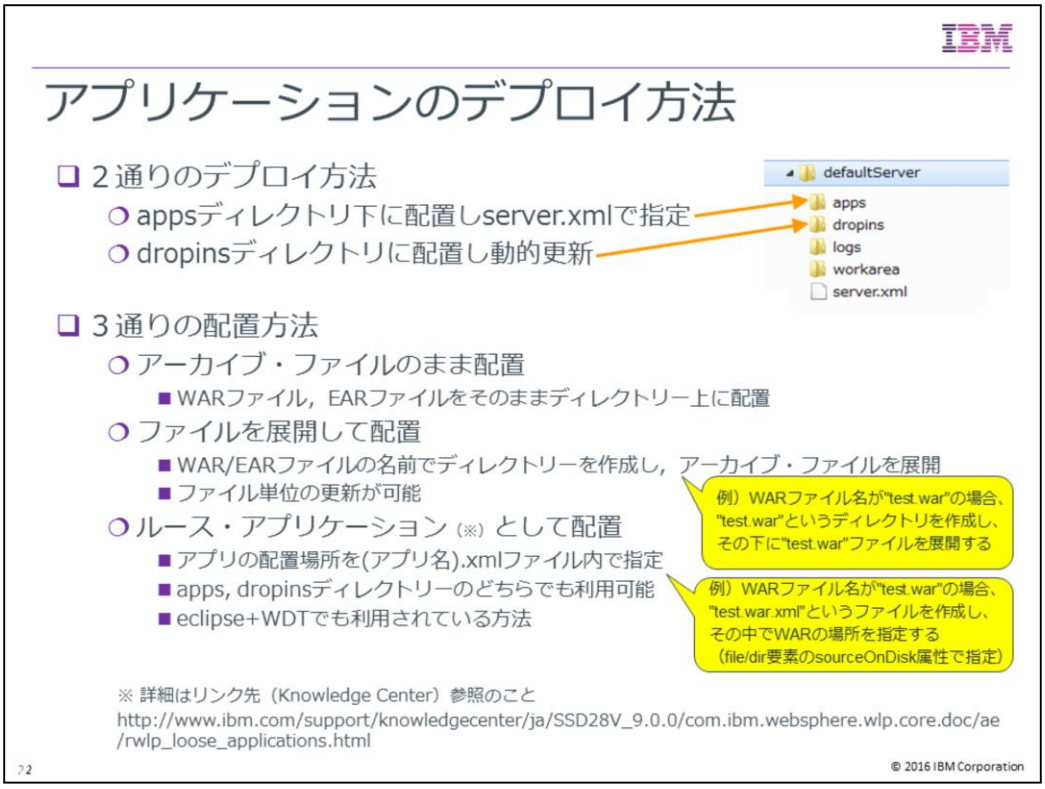
Admin Center で構成ファイルにアクセスするとき、スタンドアロン構成、Collective構成ともに、すべての構成情報ファイルにアクセスすることが可能です。



構成変更で扱うファイルは、チャートの通り、XMLである構成情報ファイル郡と、非XMLであるサーバー環境ファイルの2種類に分けられます。一般論として、XMLファイルの編集には専用のエディターを利用する方が有利です。

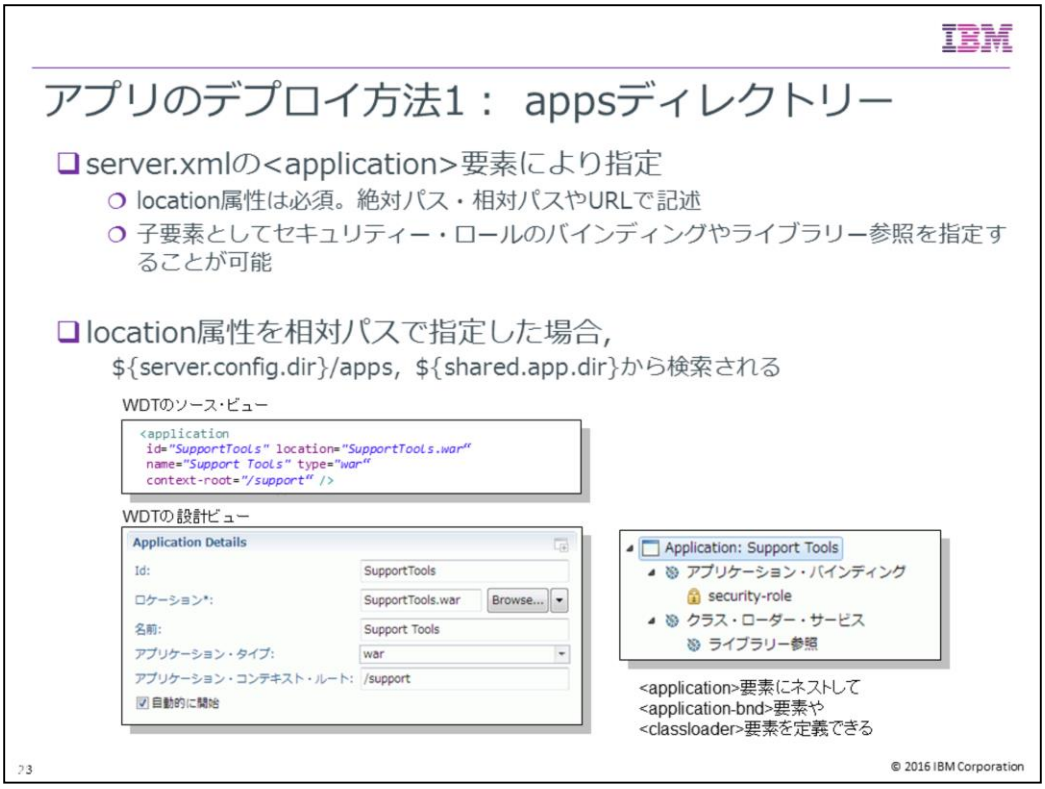
比較表では、一般のエディター、Admin Center、Eclipse+WDTの3種類を比較してます。Eclipse+WDTが全体として優れています。

アプリケーションのデプロイ



Libertyでのアプリケーションのデプロイ方法には2通りあります。１つ目は、appsディレクトリに配置して、server.xmlに<application>エレメントを記載していく方法です。もう１つは、dropinsディレクトリに配置することで、アプリケーションの導入を行う方法です。これら2通りについては、次ページ以降で説明します。

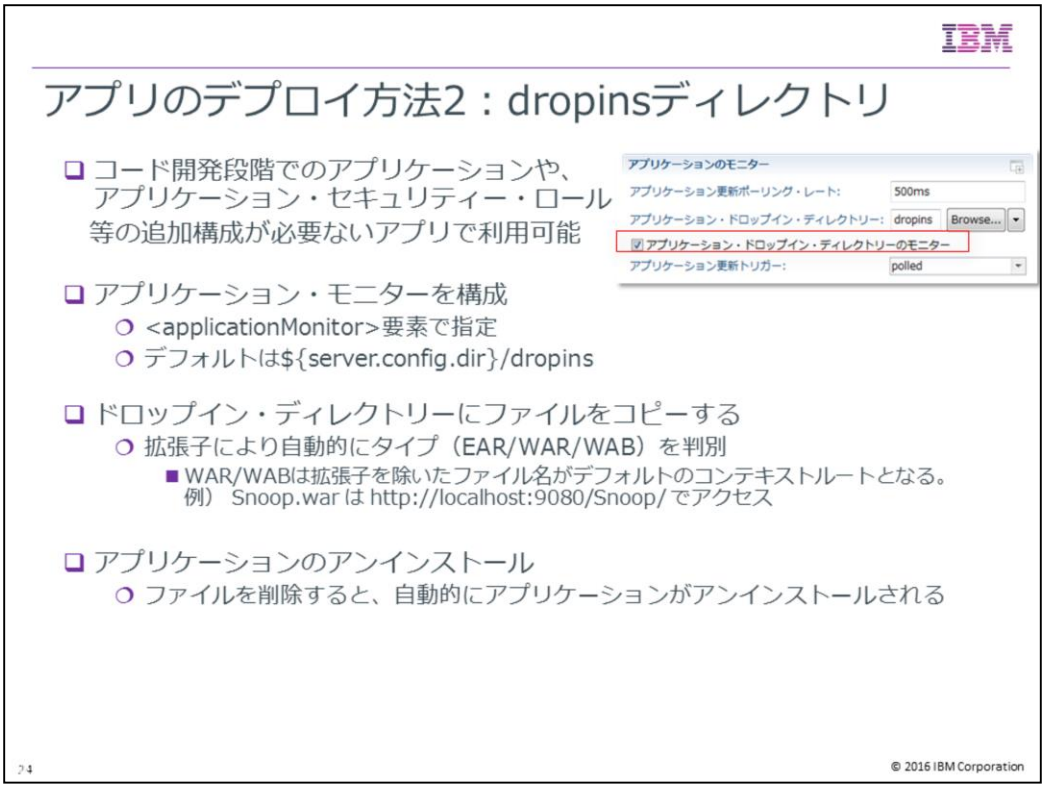
また、配置の方法としてはチャートの通り、3通りあります。



まず、appsディレクトリに配置して、<application>エレメントを指定して導入する方法です。

アプリケーションに関して、セキュリティ属性やクラスローダーや共有ライブラリの設定などが必要な場合に利用します。また頻繁な更新が発生しない本番環境ではこちらで明示的に構成を行ってアプリケーションを導入していきます。

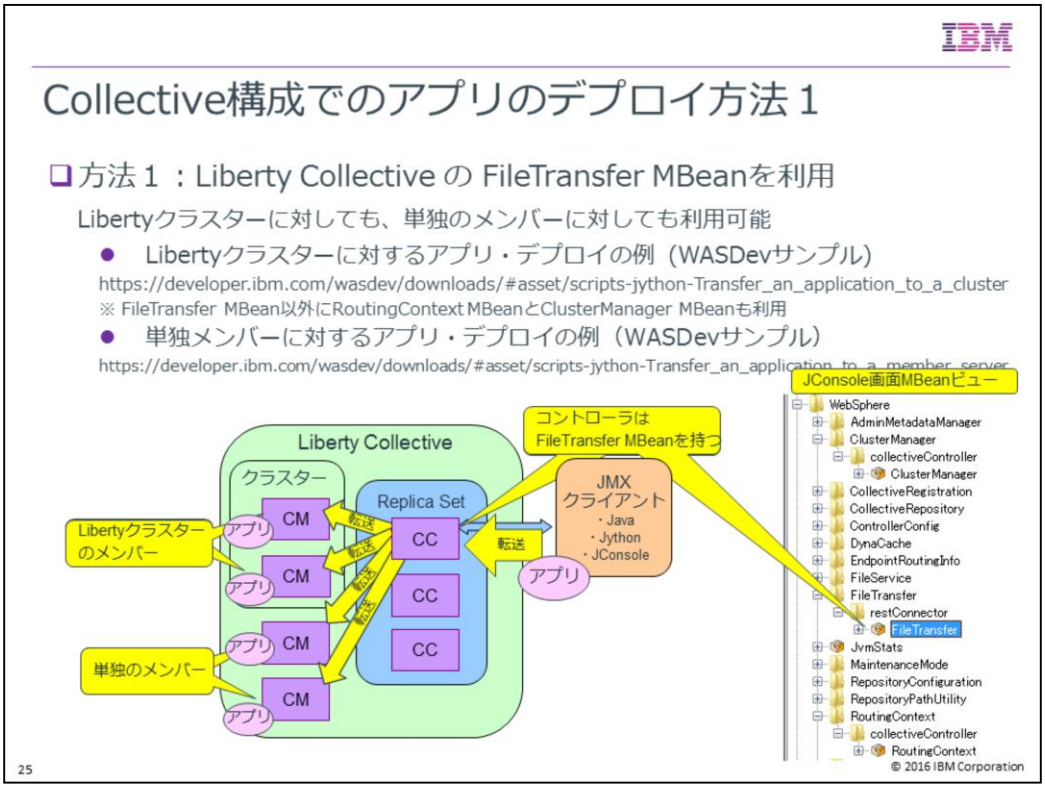
アプリケーションのロケーションを指定する際に、相対パスで指定した場合、サーバー構成ディレクトリ下のアプリケーション・ディレクトリや共有アプリケーション・ディレクトリから検索されます。



次にドロップイン・ディレクトリで指定する方法です。さきほど見たようなセキュリティーなど詳細な設定が不要な場合は、こちらを利用できます。特に開発段階などで頻繁なアプリケーションの更新が発生する場合には、こちらの導入方法が適しています。

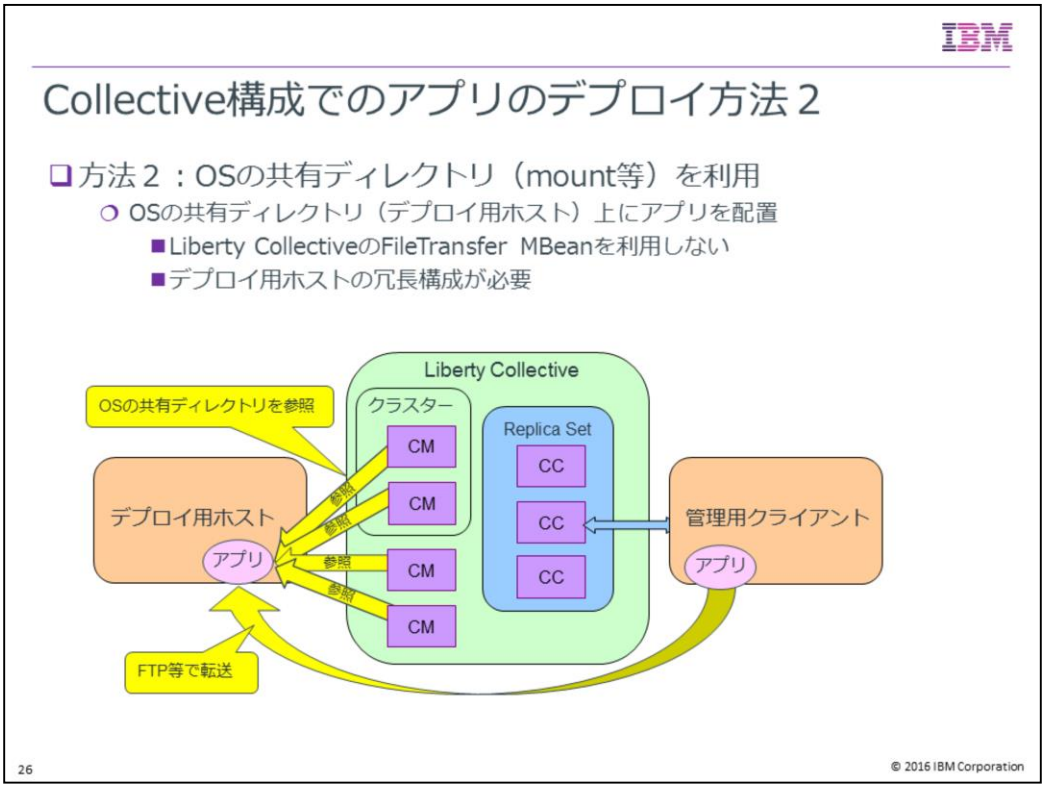
サーバーの構成の中で、アプリケーション・マネージャーというコンポーネントを有効にし、アプリケーションをドロップインするディレクトリを指定します。

ディレクトリから削除することでアプリケーションはアンインストールされます。



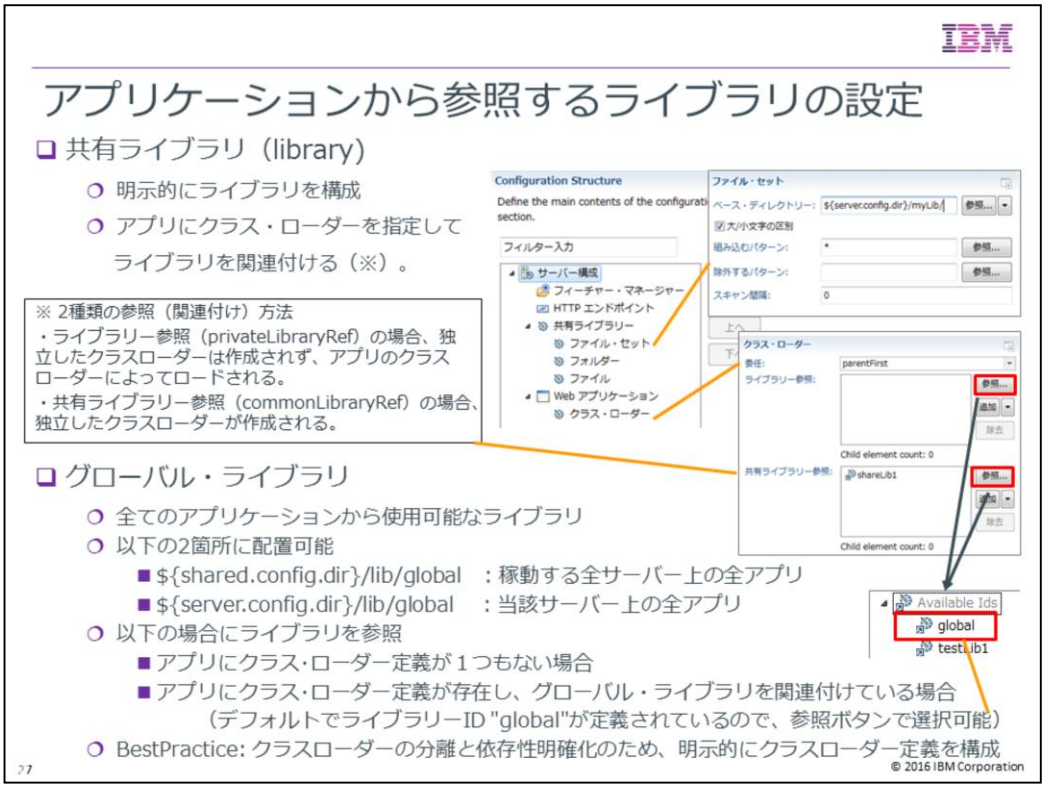
Libertyでのアプリのデプロイ方法としては、前ページまでで見てきた通りです。これに加えて、Collective構成の場合には、さらに検討すべき要素があります。

Collectiveのコントローラーでは FileTransfer MBeanが提供されているので、これを利用してJMXクライアントからコントローラーを経由して、メンバーにアプリをデプロイすることが可能です。対象となるメンバーはクラスターのメンバーでも、単独のメンバーでも可能です。詳しい方法については、チャート中のWASDevのリンク先にあるサンプルを参照してください。



Collective環境では、前ページのように、コントローラーの持つFileTransfer MBeanを利用する方法がありますが、それを利用せずに、OSの共有ディレクトリーを利用する方法もあります。

デプロイ用のホスト（サーバー）を用意して、共有ディレクトリーを作成します。これをメンバーのホストからマウントして、アプリ配置用ディレクトリーとしてserver.xml内で定義します。管理用クライアントからFTP等を利用して、デプロイ用ホストの共有ディレクトリーにアプリを配置して、メンバーにアプリを参照させます。この場合、デプロイ用ホストの冗長構成についてはインフラ側で検討する必要があります。

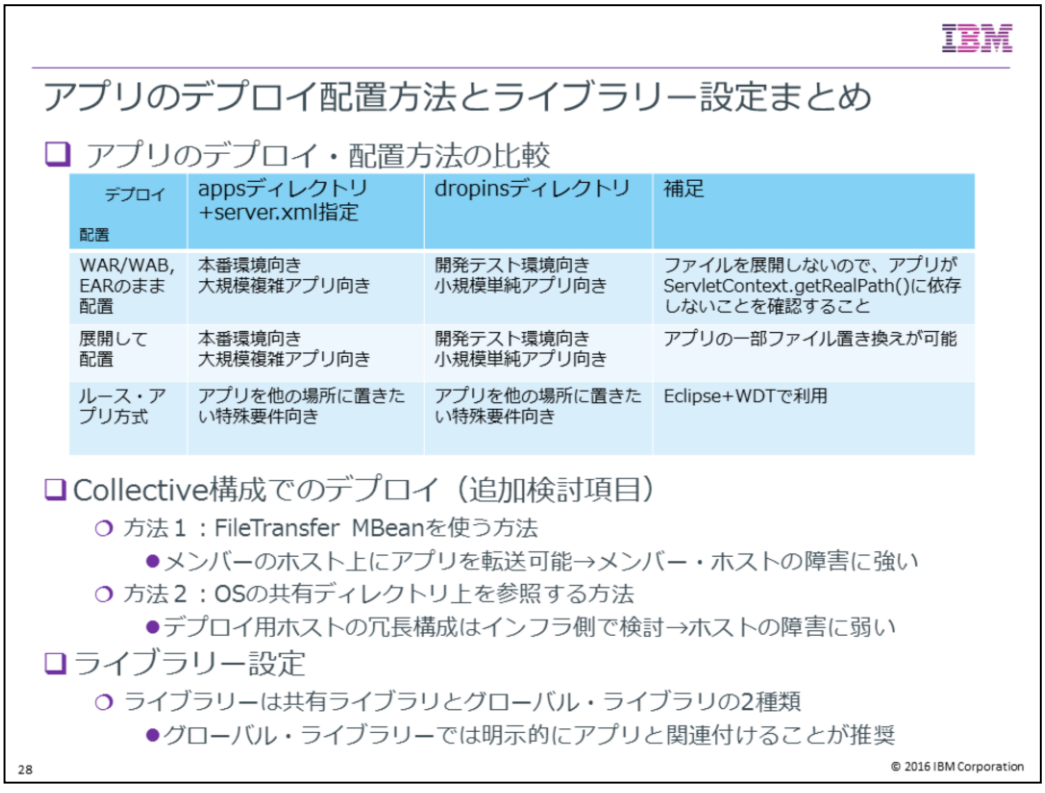


アプリケーションが参照するライブラリの管理についてご説明します。

まず共有ライブラリです。明示的にライブラリとして構成し、それを共有ライブラリとしてクラスローダーの管理画面から関連付けます。関連付ける方法（「参照」と呼ばれます）には、チャート内の枠内の通り、2種類の方法があります。

もう一つの定義方法が、グローバル・ライブラリです。これは全てのアプリケーションから使用可能となるライブラリであり、スコープにより２つの配置場所があります。稼動する全サーバー上のアプリで共有する場合は、共有構成ディレクトリ下の ${shared.config.dir}/lib/global に配置します。サーバー・インスタンス内で稼動するアプリケーション内で共有する場合には ${server.config.dir}/lib/global に配置します。

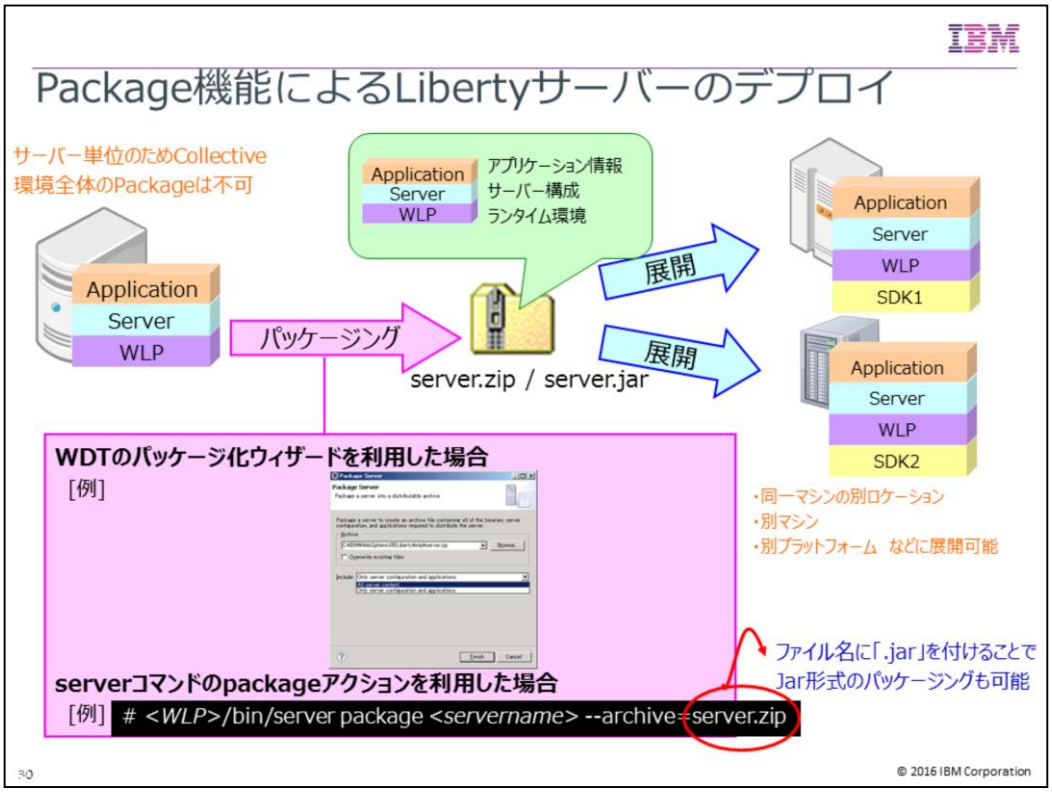
グローバル・ライブラリが利用されるための条件は、チャート内の説明の通りです。2条件のうち、後者の方法が推奨されています。アプリとライブラリーの関係が明確になるのと、参照が明確に定義できるので、クラスローダーの分離も容易なためです。なお、参照するグローバル・ライブラリは事前定義済みのライブラリーIDである「global」を指定します。



このチャートでは、アプリのデプロイ配置方法とライブラリー設定をまとめています。

表の通り、デプロイ方法については、appsディレクトリ＋server.xmlの方法が、本番環境には適していると言えます。また、この表の補足カラムの通り、アーカイブ・ファイルのまま配置する方法は、ServletContext.getRealPath()のメソッドをアプリが利用していると、ファイル・システム上の絶対パスを返すことができないため、Nullが返ります。アプリでこのメソッドを利用していないこと、このメソッドに依存するコンポーネントを利用していないかに注意してください。

Libertyサーバーのデプロイ



ここまでは、Libertyサーバー上にアプリをデプロイする方法を見てきました。ここでは、Libertyサーバー自体をまとめてデプロイする方法について説明します。

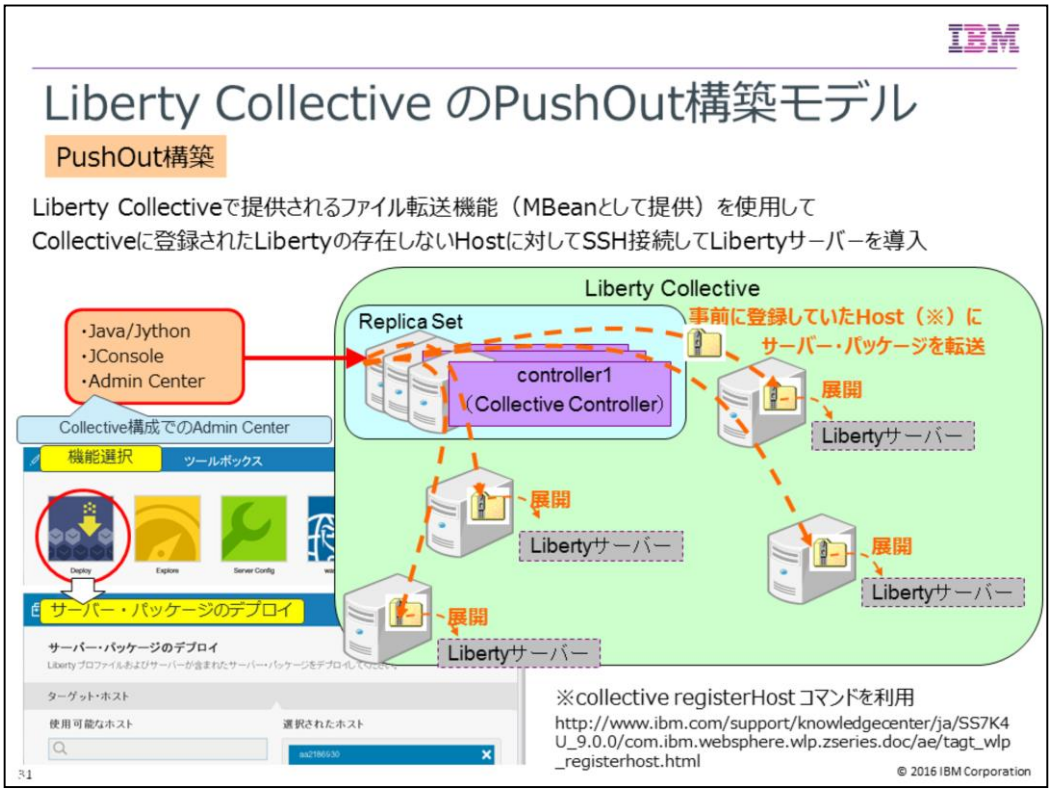
Package機能によるデプロイとは、一度構築したランタイム環境、サーバー構成、アプリケーション情報を全て1つのzip形式の圧縮ファイルにパッケージングし、それをzip展開することによってLibertyプロファイルを導入するという方法です。

パッケージングを行う機能はLibertyプロファイルが提供しており、WDTのパッケージ化ウィザードを利用したGUIベースで実行することも、serverコマンドのpackageアクションを利用したコマンドベースで実行することもできます。

zip展開先としては、パッケージングを行った同一マシンの別ロケーションでも、別マシン上でも、別プラットフォームのマシン上でも可能です。別プラットフォームのマシンにzip展開する場合、ディレクトリ名やファイル名の大文字小文字区別の要否などに関しては考慮する必要があります。

V8.5では、Package機能によりパッケージングされた後のファイルはzip形式の圧縮ファイルでしたが、V8.5.5の新機能として、実行時にファイル名に「.jar」を付けることにより、Jar形式の圧縮ファイルにすることもできるようになりました。

なお、Java SDKについてはLibertyパッケージに含めることはできないため、展開先のホストに別途用意されていることが前提となります。



前ページでは、Liberyサーバーをパッケージして、Libertyをデプロイする方法を説明しました。この方法をCollective環境で利用する場合について説明します。

Collective構成において、メンバーのLibertyサーバーをデプロイするのに、Libertyパッケージを利用することができます。ここでのメンバーは、まだLibertyがインストールされていないホストを指します。そのためには、事前に「collective registerHostコマンド」を利用して、Collective環境にホストを定義しておきます。登録する内容としては、ホストのOSユーザー情報や証明書、SSH関連情報です。

この方法は、Liberty Collective のPushOut構築モデルと呼ばれます。パッケージ・ファイルの転送には、アプリのデプロイにも利用されるFileTransfer MBeanが利用されます。

WASdevサイト(英語)ではMBeanを利用したサンプルを紹介・提供しています。

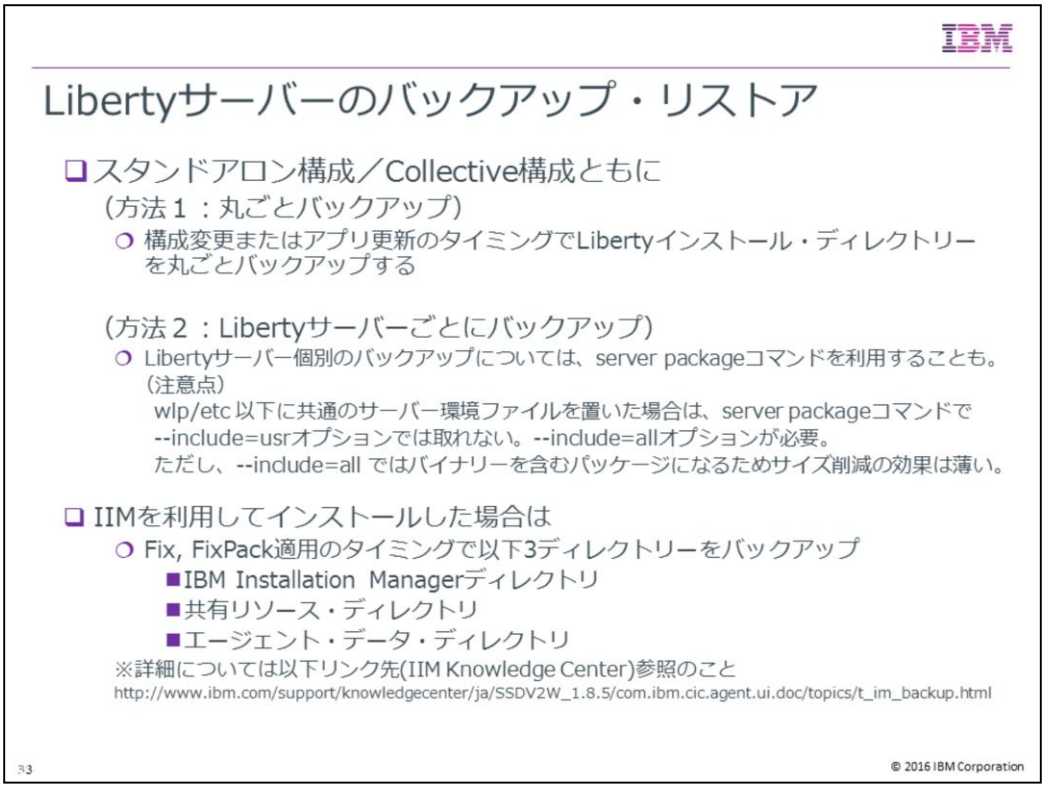
Deploy Collective Members

https://developer.ibm.com/wasdev/downloads/#asset/scripts-jython-Deploy_Collective_Members

Using File Service and File Transfer MBeans with Liberty

https://developer.ibm.com/wasdev/docs/using-file-service-and-file-transfer-mbeans-with-liberty/

バックアップ・リストア、プロセス監視

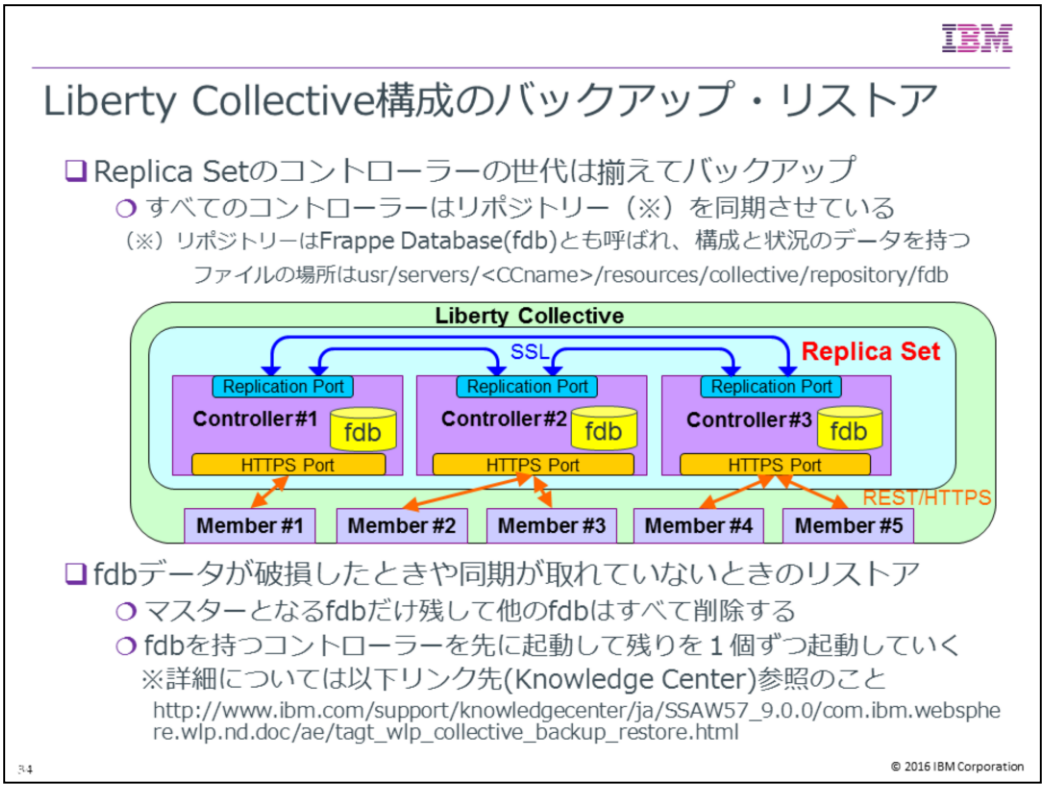


Libertyサーバーのバックアップ・リストア方法には大きく2通りの方法があります。

1つ目は、Libertyインストール・ディレクトリーの下を丸ごとtarファイル等でバックアップを取る方法です。バックアップ対象の抜けがなく、簡単な方法ですが、ファイル・サイズが大きくなるのがデメリットです。

2つ目は、Libertyのサーバーごとにバックアップを取る方法です。server package コマンドを利用することができますが、usr/servers/<server_name>の外に関連ファイルがある場合、たとえば、wlp/etcにサーバー環境ファイルを置いた場合など、--include=allオプションが必要で、1つ目の方法と比べてサイズ削減の効果は薄いかもしれません。

IIMを利用してインストールした場合は、IIM関連の3つのディレクトリーもバックアップを取る必要があります。

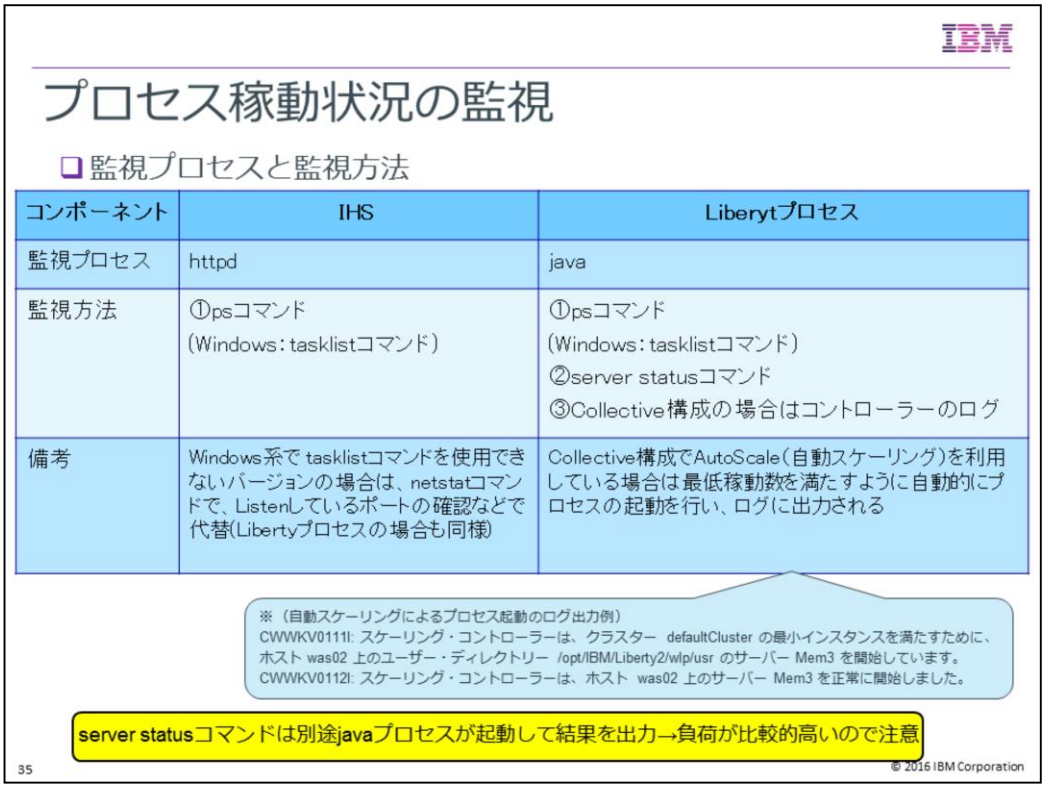


Collective構成でのバックアップ・リストアの注意点です。

Collecivの複数のコントローラーはReplica Setとして一種のクラスター構成を取っており、各コントローラーが持つリポジトリー（Frappe Databaseとも呼ばれるため、略称としてfdbが使われる）を持ちます。リポジトリーは構成情報と状況のデータを持ちます。そのパスは、チャート中の図の上にある通りです。

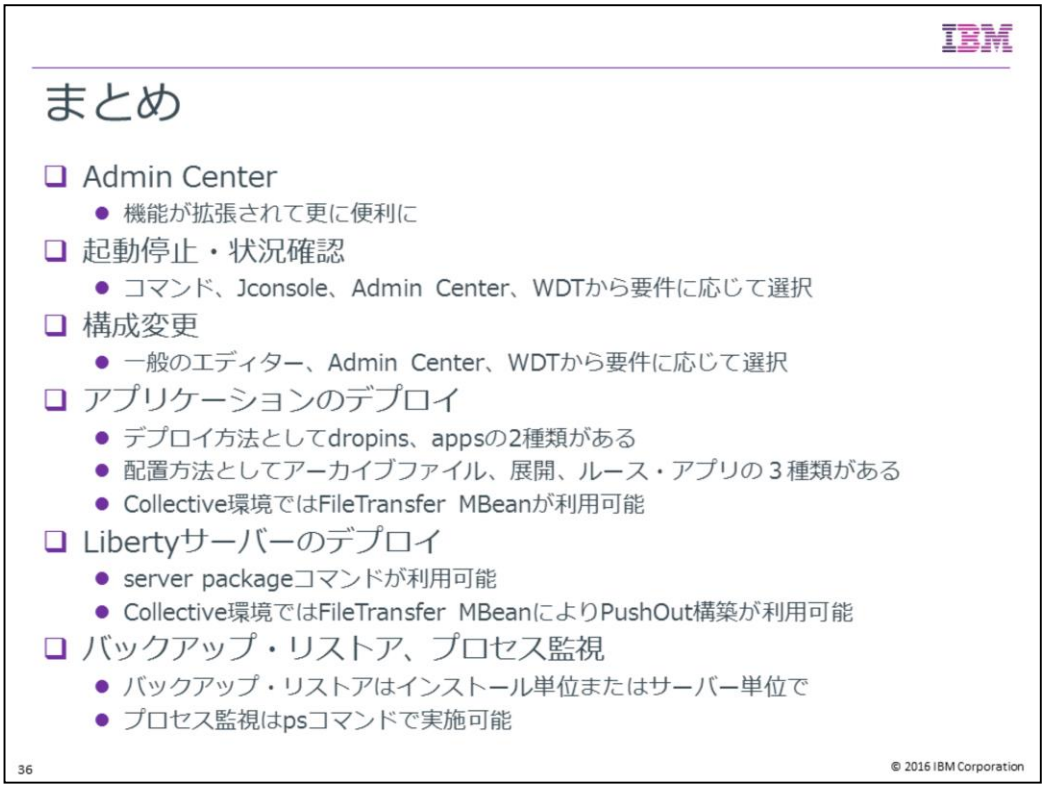
リポジトリーは常に同期を取っているため、この同期が取れていないコントローラーをReplica Setに混ぜてリストアしようとすると、正しく起動できないことがあります。そのため、構成変更やアプリやメンバーの状況が同一な状態で、全コントローラーのバックアップを取得して、リストアするときも、同じ世代のコントローラーでリストアします。

もし、リポジトリーの同期が取れていないことが判明した場合には、チャート中の一番下のリンク先の通りの手順で復活させるようにします。



上記は、一般的なIHS/WASの監視対象プロセスおよび監視方法をまとめたものです。IHSはhttpdプロセス（親プロセスのみ）、Libertyはjavaプロセスの監視を行います。一般的には、OSのpsコマンドを使用して、プロセス名やアプリケーション・サーバー名をキーにして、出力を確認します。Libertyが提供するコマンド行ツールのserver　statusコマンドでもプロセスの稼動状況を確認することができますが、このsコマンドを実行した場合には、別途javaプロセスが起動し、psコマンドの実行と比較して負荷が高くなりますので、ご注意ください。また、一部Windowsのバージョンでは、tasklistコマンドが使用できませんので、netstatコマンドで、Listenしているポート番号（IHSであれば通常80番、Libertyのデフォルトは9080番）を監視することなどで代替できます。

Collective構成で自動スケーリングを利用している場合には、プロセスの数が最低稼動数を満たすように自動的にLibertyメンバーが起動されます。その場合には、チャート中の噴出し部分のようなメッセージがログに出力されます。このメッセージIDをモニターするとプロセス監視に役立つ場合があります。



このセッションの全体のまとめです。

WebSphere Application Server Liberty