IBM

WebSphere Application Server V9

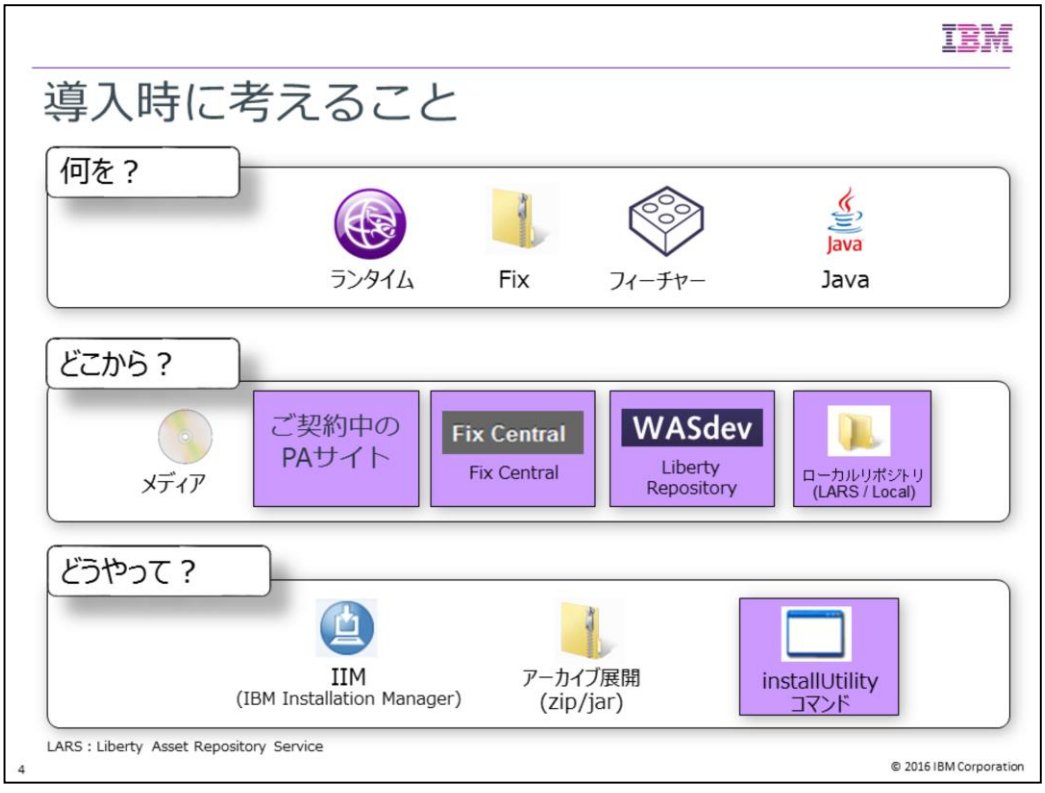
Liberty 基盤設計セミナー

導入 / 構成

アジェンダ

- 導入
  - ランタイムの導入方法
    - IBM Installation Manager / アーカイブ展開
  - 実働環境への導入作業の流れ
    - ランタイム・Java / Fix / フィーチャー
  - BluemixにおけるLibertyの導入
- 構成
  - 構成要素
    - ディレクトリ構造
    - 使用可能な環境変数
    - 構成ファイル（server.xml他、include可能、などの特徴）
  - 構成 の おまけ機能
  - 「構成」の運用を考える
    - サーバー構成をアーカイブする
    - ディレクトリ構成運用を考える
    - サーバー構成ファイルの運用を考える
  - 構成の流れ と 構成手順
    - 1. スタンドアロン構成
    - 2. シンプル・クラスター構成
    - 3. 高可用性構成

導入

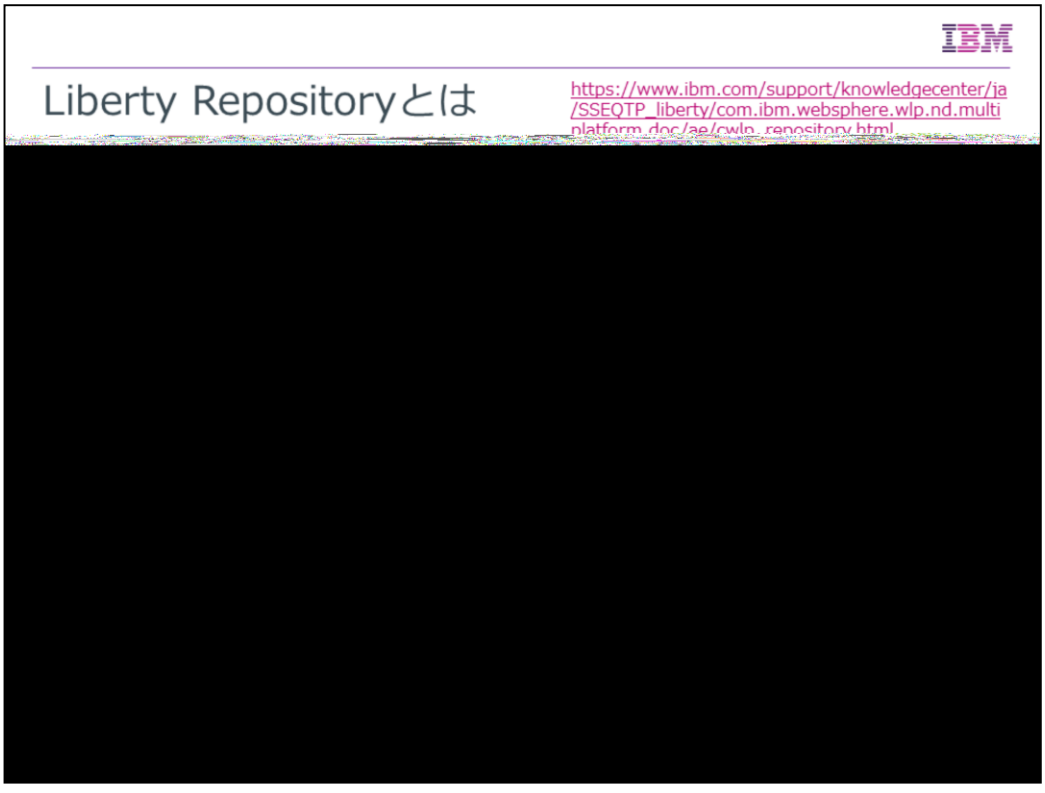


Libertyランタイムを導入する場合に、何を？どこから取得？どうやってインストール？など、考えるポイントがあります。

何を？　　：ランタイム自身、Fix、フィーチャー、ランタイム稼働に必要なJavaなどがあります。

どこから？　：メディアDVD、PAサイト、Fix Centralがあります。これとは別にWASdev(Liberty Repository)というWebサイト、オフラインアクセス可能なローカルリポジトリ(LARS/Local)から取得します。

どうやって？：IIM(IBM Installation Manager)、アーカイブ展開(zip/jar)、installUtilityコマンドを利用します。



Liberty Repositoryは、Libertyに関連する様々なアセットがダウンロード可能なリポジトリです。

基本的には、WASdev(要インターネットアクセス)を指します。

オフラインで使用する場合には、LARSやLocal Repositoryを事前に構築しておくことで対応できます。

取得可能なアセットは、アドオン、管理スクリプト、構成スニペット、フィーチャー、オープンソース統合、製品、製品サンプル、ツールがあります。

WASdevのサイトには以下のURLでアクセスできます。

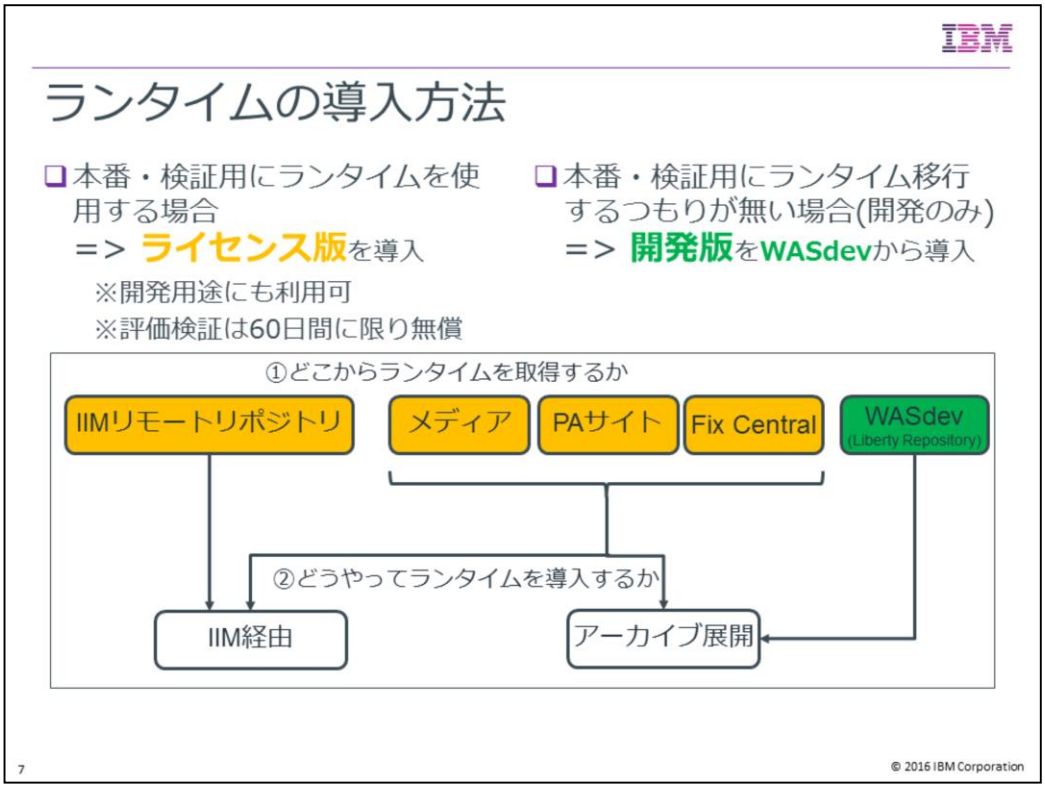
https://developer.ibm.com/wasdev/

前スライドであげたLibertyに関連する様々なアセットがダウンロード可能です。

フィーチャーに関しては、Libertyの導入ディレクトリのbin以下にあるinstallUtilityコマンドを使用してインストールする必要があります。

また、GA版のランタイムだけでなく、月1の頻度でリリースされるBeta版も取得可能です。

Beta版には、新機能が含まれますので、いち早く確認することが可能です。



ランタイムを導入する際のモチベーションには2つあると考えています。

ひとつは、本番環境や検証環境をサーバーに構成する場合です。

導入したランタイムはそのまま本番・検証で利用しますので、ライセンス版が必要になります。

取得元は、IIMリモートリポジトリ、メディアDVD、PAサイト、Fix Centralです。

導入方法は、IIM経由およびアーカイブ展開です。

ふたつめは、開発用に導入したランタイムを本番・検証環境にそのまま移行しない場合です。

あくまで開発用途でローカル開発PCでの使用になるため、WASdev(Liberty Repository)からアーカイブ版を取得することになります。

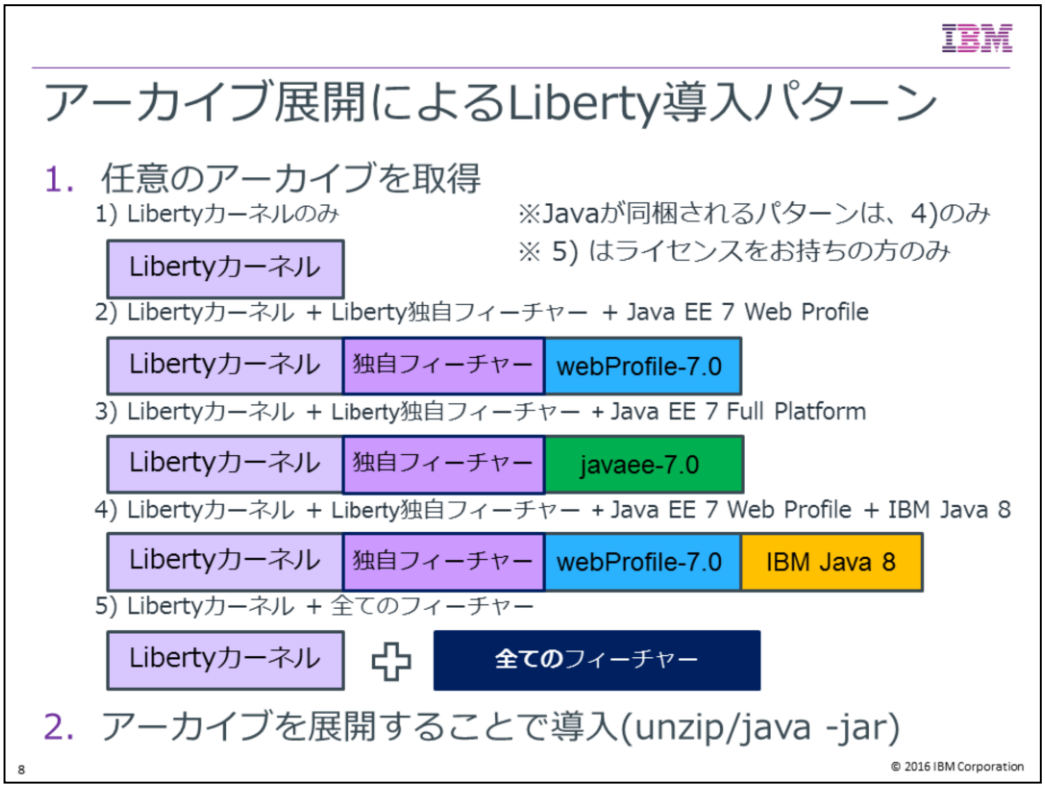
※後のスライドで述べますが、開発用途無償版(BASE_ILAN、サポート無し)を、本番・検証環境として移行することも可能です。その場合、ライセンス化ファイルを適用することでライセンス版にアップグレードすることができます。

取得元は、WASdevやFix Centralです。

導入方法は、アーカイブ展開です。

IIMでインストールするためのイメージを取得した場合は、IIM経由でLibertyを導入します。

アーカイブ形式で取得した場合は、アーカイブ展開で導入することになります。

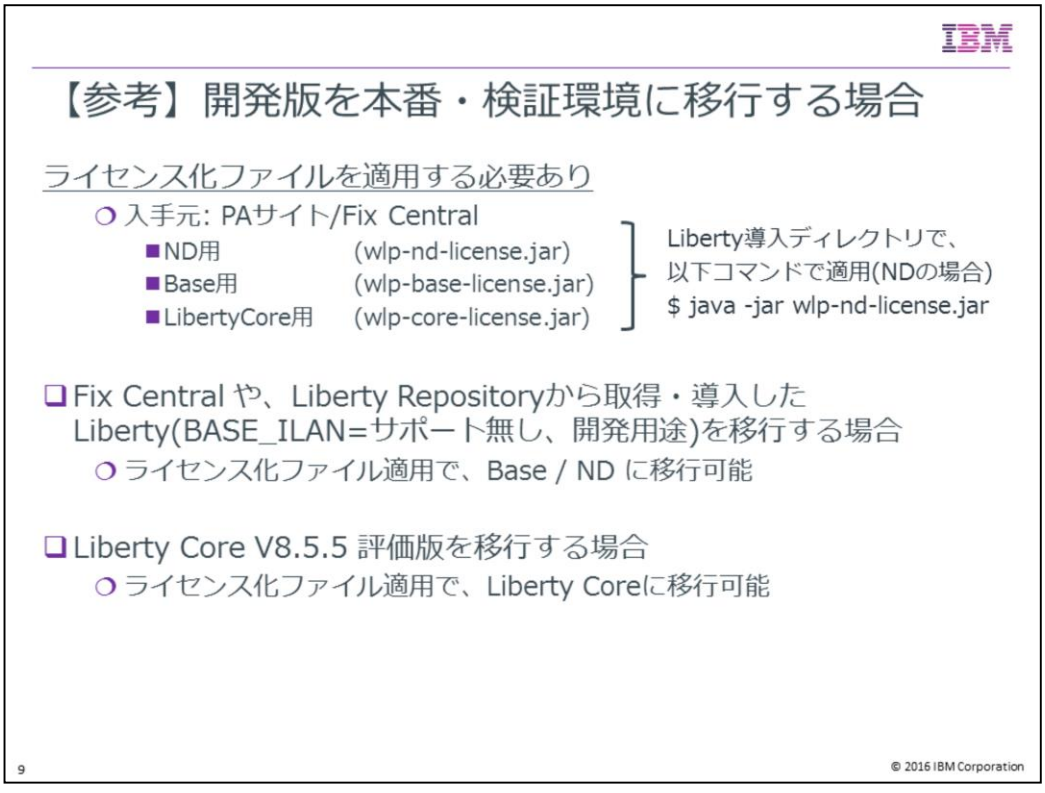


アーカイブ展開によるLiberty導入パターンについてです。
手順はシンプルで、任意のアーカイブを取得し、unzipやjava –jarで展開するだけです。

1-4はzip形式で提供されるもので、ランタイム機能のみのLibertyカーネルや、Java EEのフィーチャーやLiberty独自フィーチャーが含まれるものがあります。
5は、jar形式でライセンス版のみ提供されるもので、全てのフィーチャーが導入済のアーカイブになります。

【アーカイブ(zip/jar)の入手元】
開発用途無償版は、
Liberty Repository、または、Fix Centralから入手可能です。
zip形式のアーカイブが全て含まれます。

ライセンス版のjar形式のアーカイブは、
Liberty Core、Base、NDすべての製品エディション用に提供しています。
メディア、PA、Fix Centralから入手可能です。
webProfile-6.0が含まれるjarと、フィーチャーが全て導入された.jarがあります。
※フィーチャー全部入りのjarは、Fix Centralからのみ入手可能です。



開発用途無償版(BASE_ILAN、サポート無し)として導入したLibertyを、本番・検証環境に移行する際には、ライセンス化ファイルを適用する必要があります。

ライセンス化ファイルの適用により、ライセンス版にアップグレード可能です。

【ライセンス化ファイルの入手元】

ND/Base/Liberty Coreの各エディション用のライセンス化ファイルは、PAサイトもしくはFix Centralから入手可能です。

【ライセンス化ファイルの適用】

Liberty導入ディレクトリで、java -jarコマンドを使用して適用します。

(例: java –jar wlp-nd-license.jar)

【注意点】

Liberty Repositoryなどから入手した開発用途無償版(BASE_ILAN、サポート無し)に対して、Liberty Coreへのアップグレードはできません。

Liberty Core用のライセンス化ファイルは、Liberty Core評価版(V8.5.5)を本番移行する際に使用します。

IIM経由でLibertyを導入する場合は、IIMでリポジトリの登録を行います。その方法は2つあります。

メディアやPAサイト、Fix Centralなどから取得したイメージを指定する方法と、リモートリポジトリを指定する方法です。

後者のリモートリポジトリを指定する場合のURLを以下に示します。

- NDエディション

http://www.ibm.com/software/repositorymanager/V9WASND

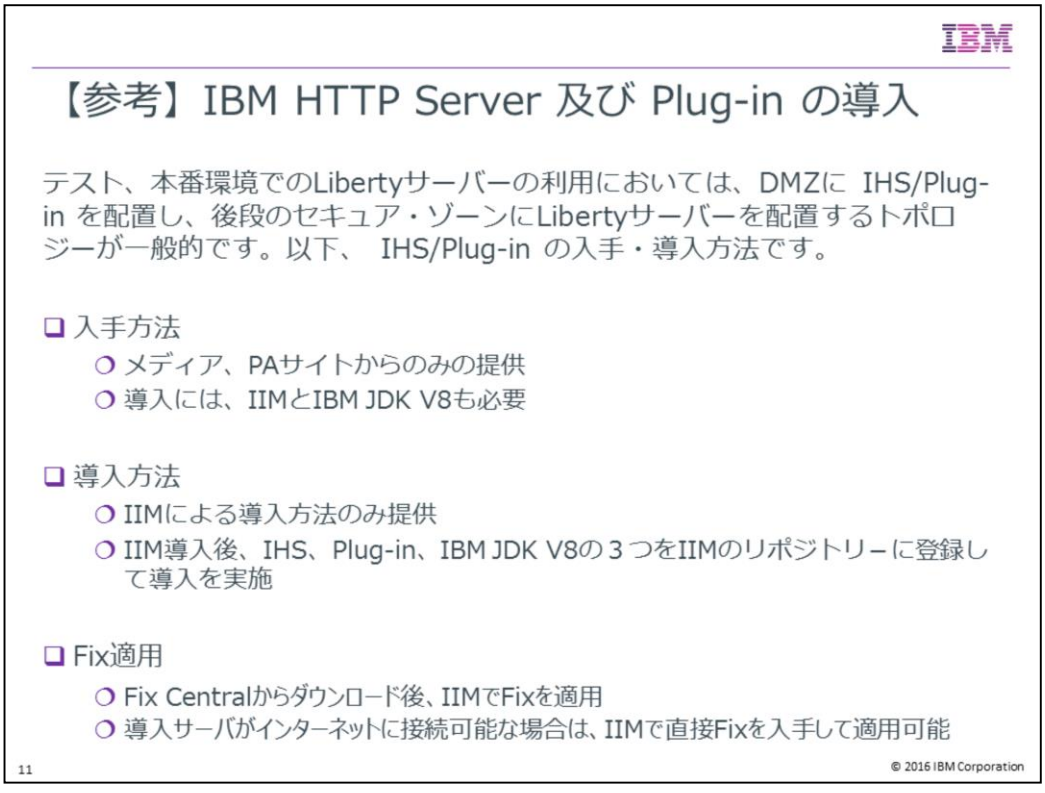
- Baseエディション

http://www.ibm.com/software/repositorymanager/V9WASBase

※いずれも、本番もしくは検証(60日間)の利用用途のためのエディションになります。

導入時には、フィーチャーやアドオン、サンプルなどを選択可能です。

フィーチャーについては、全部入りもしくは個別選択が可能です。

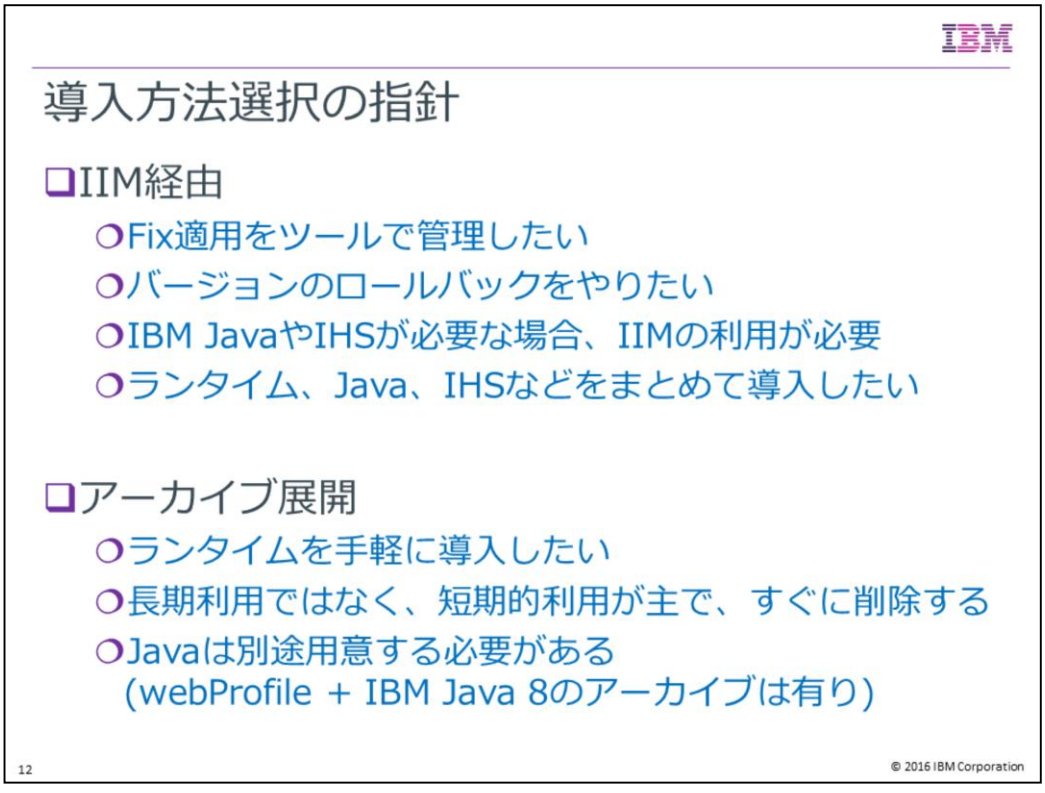


IHSやPlug-inを導入する際には、IIM経由での導入が必須となります。また、IBM Javaの導入も必須となります。

Libertyの導入時と同様に、IIMでリポジトリの登録を行います。その方法は2つあります。
メディアやPAサイトから取得したイメージを指定する方法と、リモートリポジトリを指定する方法です。

リモートリポジトリは以下です。
http://www.ibm.com/software/repositorymanager/V9WASSupplements

導入後のFix適用は、IIM経由で行います。



Libertyの導入方法は、IIM経由とアーカイブ展開の2パターンがあります。
本スライドでは、Libertyを導入する際の指針について説明します。

満たしたい要件にあった方を選択してください。

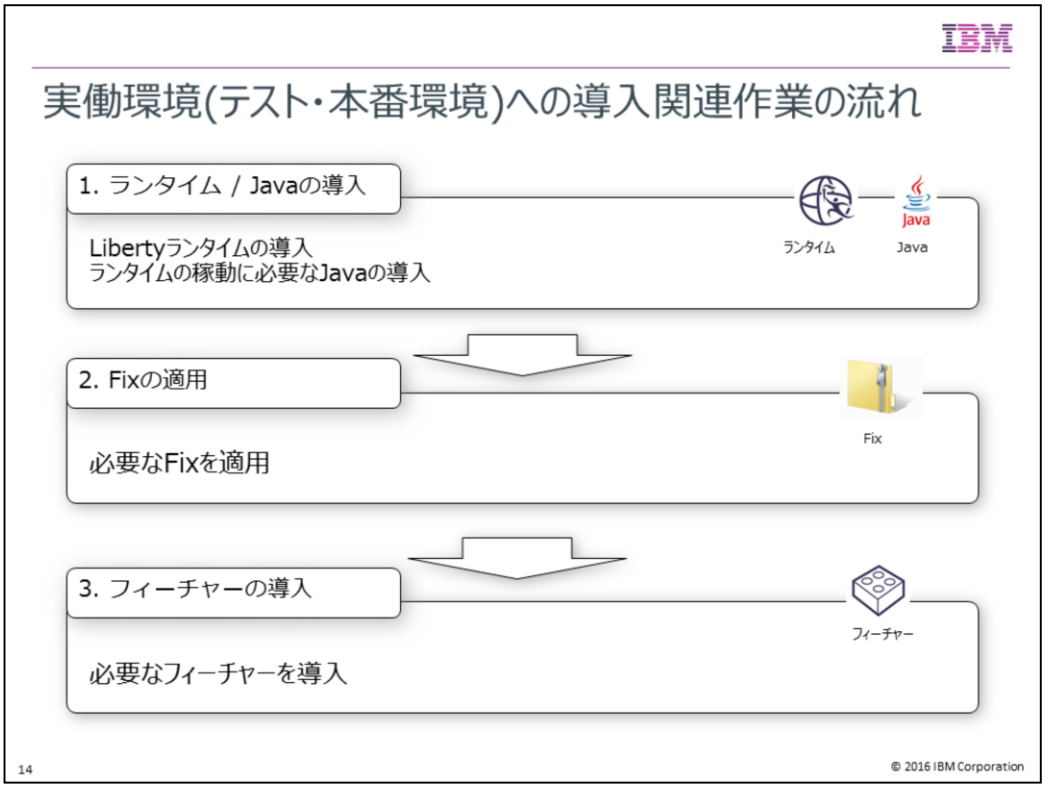
【IIM経由で導入する】
単一のツールで、必要となるリポジトリの指定によりFix適用や、ロールバックが行えます。このため、Libertyの導入だけでなく、その後のFix適用も含めてツールで管理したい場合に有効です。
また、IBM JavaやIHS、Plug-inなどが必要な場合、これらはIIM経由での導入が必須となります。このため、Libertyを含めて単一のツールのみで導入できるメリットがあります。

【アーカイブ展開】
アーカイブを展開するだけであるため、手軽に、スピーディーに導入したい場合に有効です。
例えば、一部のユーザーに対して実施するトライアル(Beta期間)など短期間での利用の場合にも有効です。
※ただし、IBM Javaは別途導入する必要があります。IHSやPlug-inなどが必要な場合も別途IIM経由での導入が必要になります。(webProfileフィーチャーと共にIBM Java 8が同梱されたアーカイブはあります)

【参考】導入方法による差異 (IIM経由 or アーカイブ展開)

	Installation Manager	アーカイブ展開
エージェントなしでの導入	No	Yes
任意のFixPackレベルに直接上げられるか？	Yes	Yes
FixPackの上書き導入	Yes	No
Interim fixの上書き導入	Yes	Yes
FixPack/iFixのロールバック	Yes	No
製品のエディションのアップグレード	Yes	Yes
組み込み SDK 提供	Yes	Yes
developer toolsとの統合	No	Yes
フィーチャーの追加	Yes	Yes
Liberty Repository の統合	Yes	Yes
z/OS パッケージの提供	Yes	No
パッケージ Minification のサポート	Yes （生成されたイメージは新しいアーカイブとして扱われ、IIMの管理対象にはなりません）	Yes

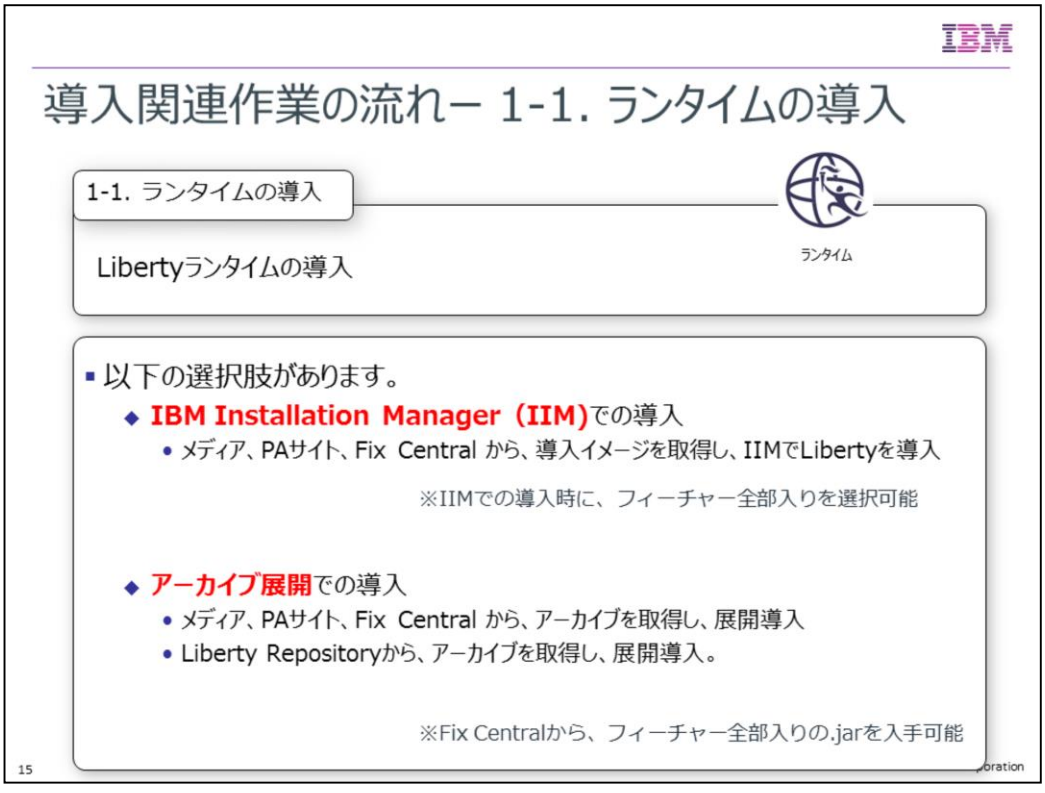
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_inst_top.html



実働環境へのLiberty導入関連作業についての基本的な流れは以下です。

1. ランタイム および Javaの導入
2. Fixの適用
3. フィーチャーの導入

最初に、Libertyを導入する際の方法によって、後続の手順が変わってきます。



【1-1.ランタイムの導入】

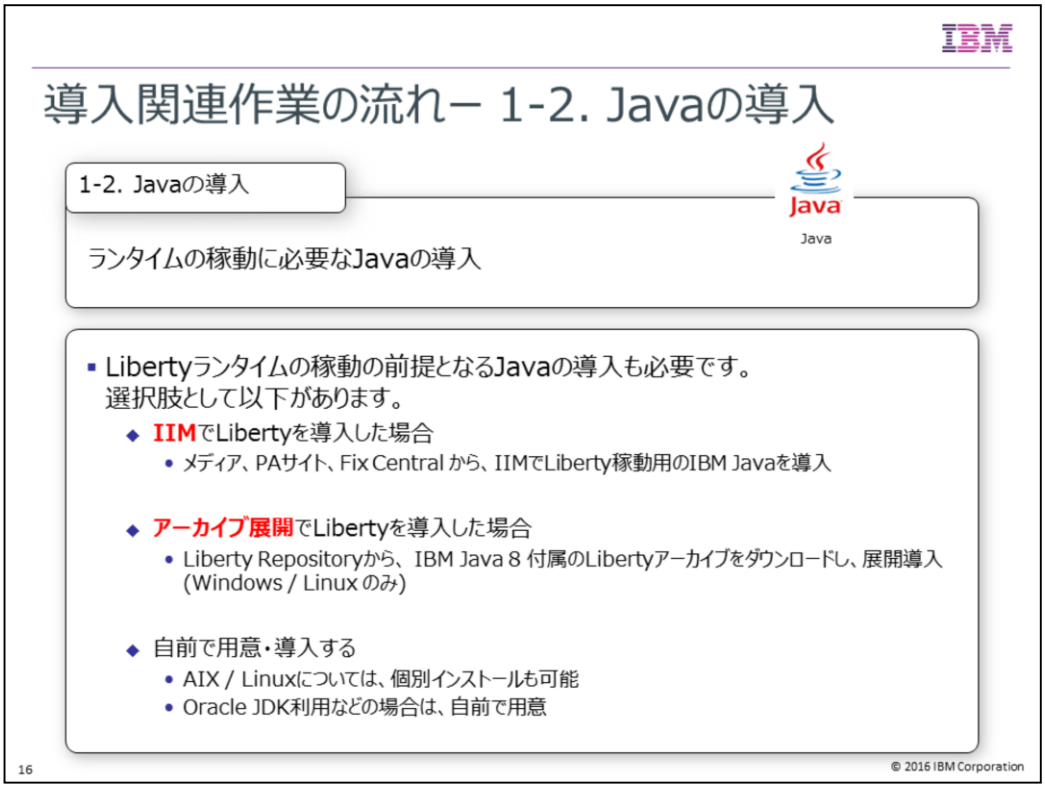
2パターンがあります。

・IIM経由での導入

・アーカイブ展開での導入)

それぞれフィーチャー全部入りの状態で導入することも可能で、最初に行っておくことで後の作業が楽になります。

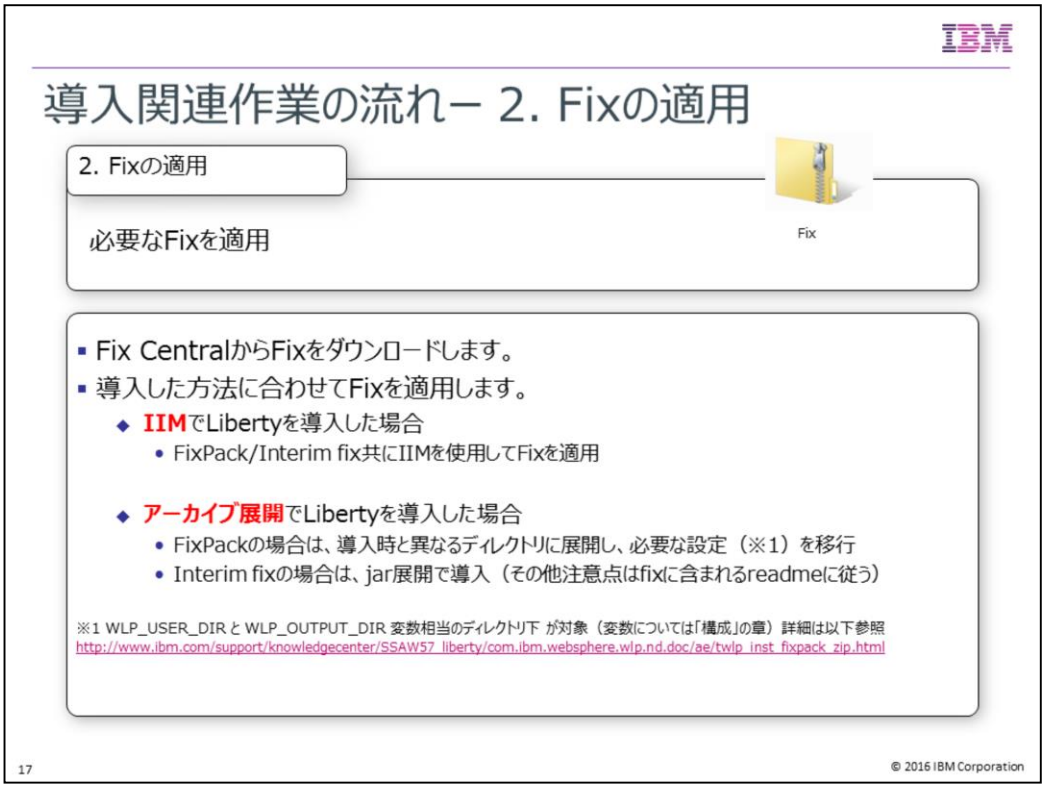
※アーカイブ展開の場合、フィーチャー全部入りの.jarがありますが、ライセンスをお持ちの場合のみに取得可能なオプションとなります。



【1-2.Javaの導入】
Libertyを稼働させるために必要となるJavaの導入です。

導入方法は3パターンあります。
・IIM経由
・アーカイブ展開(webProfileのみ)
・自前のJavaを利用

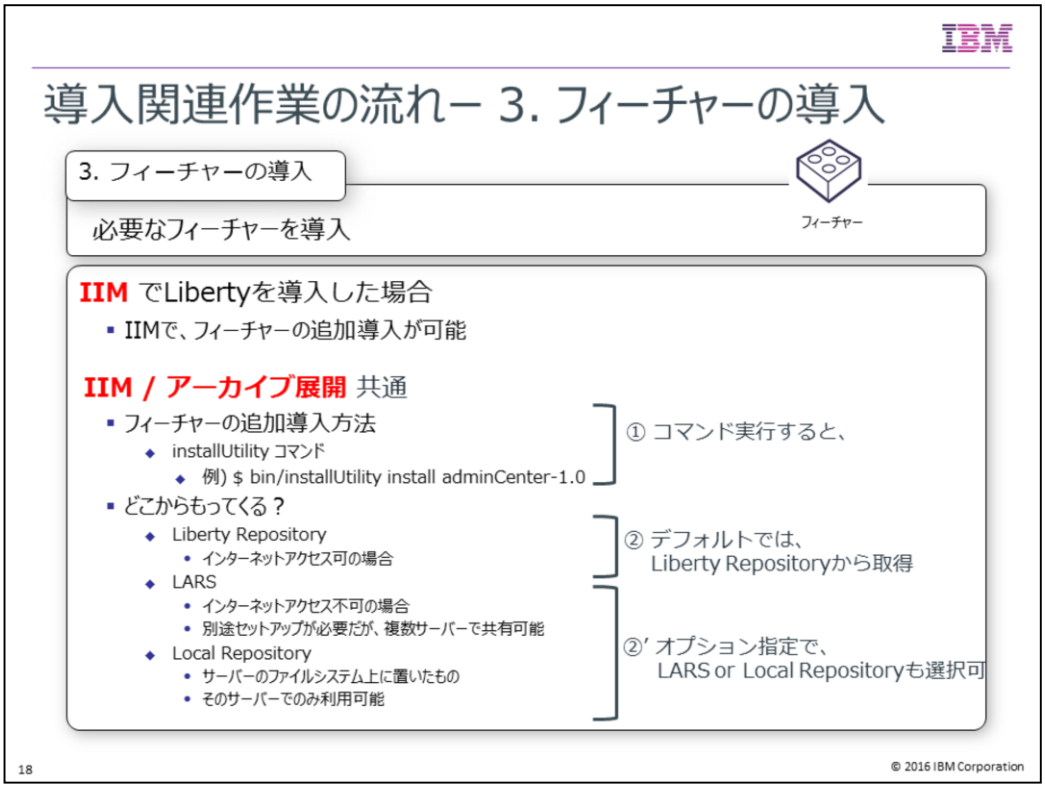
Libertyは、IBM Javaだけでなく、Oracle JDKも利用可能です。(ただしJavaのサポートが必要な場合は、IBM Javaを使用してください。)



【2.Fixの適用】
Fixの適用には、2パターンあります。
・IIM経由
・アーカイブ展開

IIM経由でLibertyを導入した場合は、
Fixpack/Interim fix共にIIM経由でのFix適用が必須になります。

アーカイブ展開でLibertyを導入した場合は、
元の導入ディレクトリとは異なるディレクトリに新しいFixpackを展開します。
すなわち、新規にLibertyを導入する形式になりますので、既存の構成を再利用する場合には必要な設定の移行が必要です。
(後半の「構成」セッションを参照)



【3.フィーチャーの導入】

Libertyの導入時に、フィーチャー全部入りを選択することで、本手順を省力化することが可能です。

フィーチャーを追加導入する際には、2つのパターンがあります。

・IIM経由での追加導入

・コマンドによる追加導入

IIM経由での追加導入は、

IIM経由でLibertyを導入した場合のみに利用できます。導入時と同様のウィザードで、フィーチャーの追加導入が可能です。

コマンドによる追加導入は、

IIM経由、アーカイブ展開のいずれの方法でLibertyを導入した場合でも利用できます。

スライド上に示しているinstallUtilityコマンドで任意のフィーチャーを指定し、導入することが可能です。

取得元は、3つあります。

・Liberty Repository (デフォルト、インターネットアクセスが必要)

・LARS (別途、用意が必要)

・Local Repository (別途、用意が必要)

【参考】 installUtilityとは

- Libertyにアセットの導入、削除、検索ができるコマンド
- コマンドの構文　　installUtility *action [options]*
  - download
    - リポジトリからアセットをローカルのファイルシステムに保存
  - find
    - リポジトリからアセットを検索
  - install
    - リポジトリにあるアセットをLibertyにインストール
  - testConnection
    - リポジトリに接続できるか確認
  - uninstall
    - Libertyに導入済のアセットをアンインストール
  - viewSettings
    - リポジトリの設定を確認
    - リポジトリの設定は、${wlp.install.dir}/etc/repositories.properties ファイルに定義

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_command_installutility.html

【参考】Liberty Repositoryからのアセット導入方法

導入方法 / アセット	installUtility コマンド	featureManager コマンド	configUtility コマンド	Installation Manager	WDT (WebSphere Developer tools)	WASdev site (Liberty Repository)
アドオン	✓	✓		✓	✓	✓
管理スクリプト						✓
構成スニペット			✓		✓	✓
フィーチャー	✓	✓		✓	✓	
オープンソース統合	✓			✓	✓	✓
製品				✓	✓	✓
製品サンプル	✓			✓	✓	✓
ツール					✓	✓

IBM

【参考】LARS と Local Repository

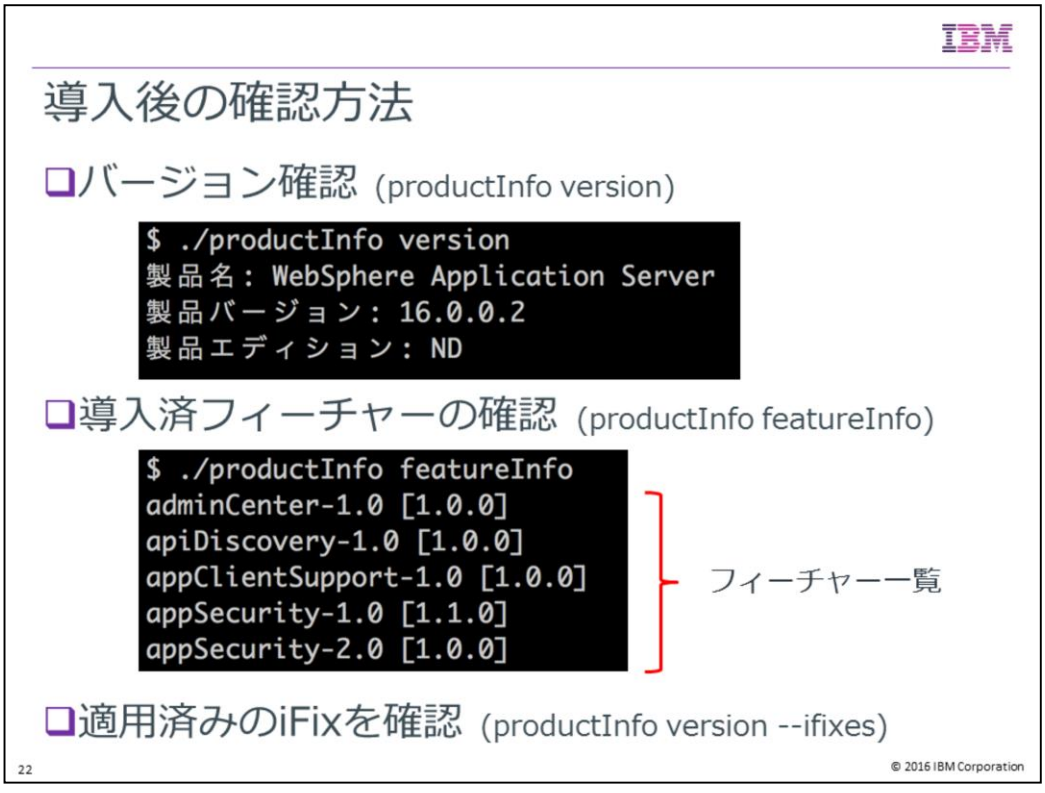
ローカルリポジトリとして以下を構成して利用できます

- LARS (Liberty Asset Repository Service)
  - Liberty Repository相当のものをオンプレミス環境に構築可能
  - 社内環境など、Internet越しにLiberty Repositoryにアクセスできない環境において有用
  - 別途導入・構築する必要あり

https://github.com/WASdev/tool.lars

- Local directory-based Repository (Local Repository)
  - ローカルマシンのファイルシステムに直接アセットを配置
  - ファイルシステムへのアクセスになるので、そのマシン上でのみ利用可能
  - installUtility downloadコマンドで作成可能

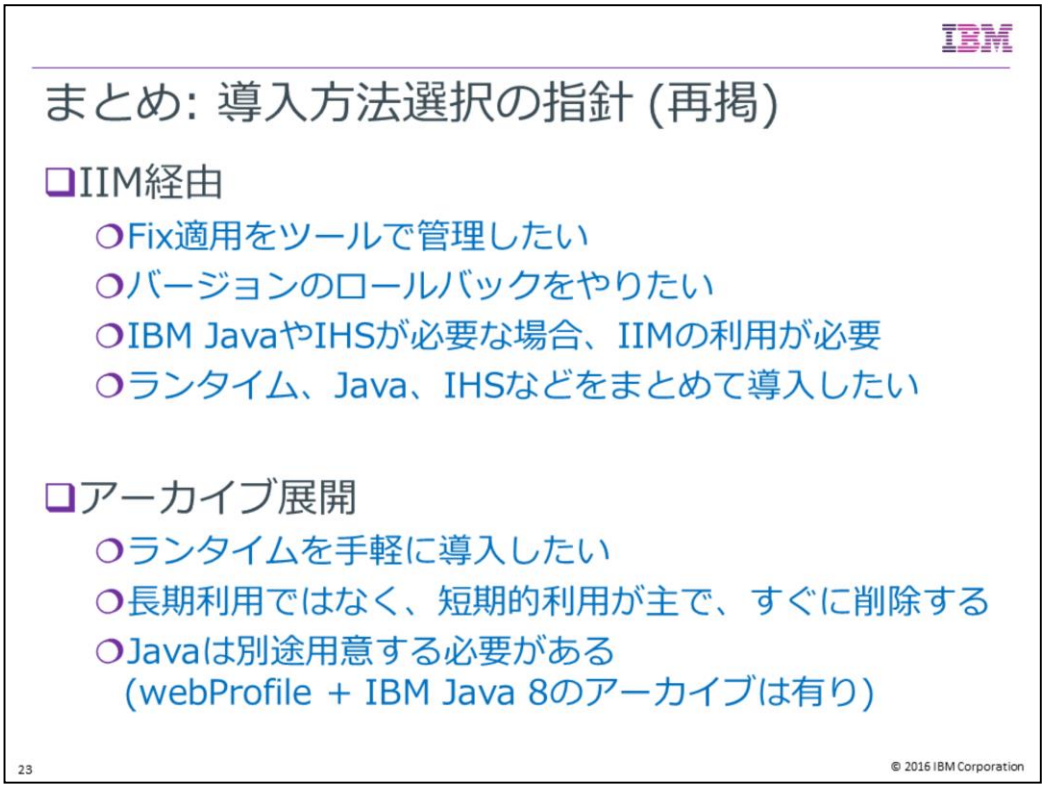
https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/twlp_ins_installUtilitydl.html



Liberty導入後の確認方法についていくつか示します。
・製品名・バージョン・エディションを確認
bin/productInfo version

・導入済のフィーチャー一覧を確認
bin/productInfo featureInfo

・適用済iFixを確認
bin/productInfo version --ifixes



(再掲)

Libertyの導入方法は、IIM経由とアーカイブ展開の2パターンがあります。

本スライドでは、Libertyを導入する際の指針について説明します。

満たしたい要件にあった方を選択してください。

【IIM経由で導入する】

単一のツールで、必要となるリポジトリの指定によりFix適用や、ロールバックが行えます。このため、Libertyの導入だけでなく、その後のFix適用も含めてツールで管理したい場合に有効です。

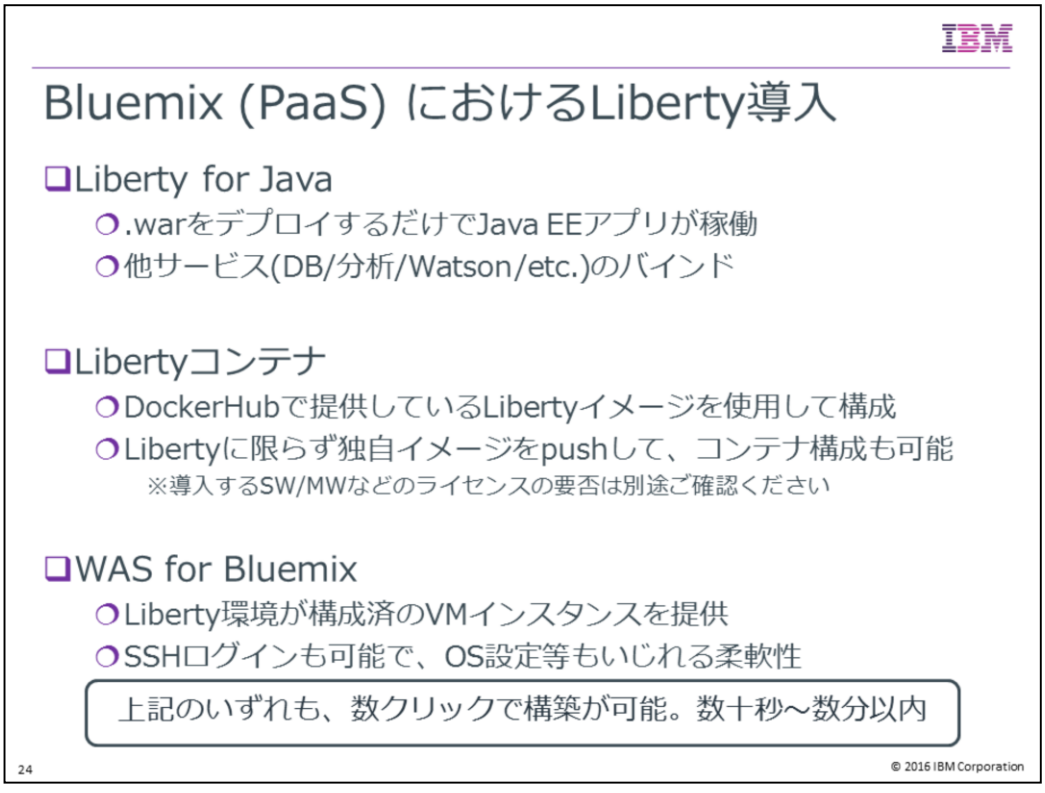
また、IBM JavaやIHS、Plug-inなどが必要な場合、これらはIIM経由での導入が必須となります。このため、Libertyを含めて単一のツールのみで導入できるメリットがあります。

【アーカイブ展開】

アーカイブを展開するだけであるため、手軽に、スピーディーに導入したい場合に有効です。

例えば、一部のユーザーに対して実施するトライアル(Beta期間)など短期間での利用の場合にも有効です。

※ただし、IBM Javaは別途導入する必要があります。IHSやPlug-inなどが必要な場合も別途IIM経由での導入が必要になります。(webProfileフィーチャーと共にIBM Java 8が同梱されたアーカイブはあります)



IBMのクラウド環境であるBluemixでは、3パターンの形でLiberty環境を構築できます。

以下に紹介するBluemixサービスを活用することで、クラウド上で、迅速に開発・検証・サービス提供を行うことが可能です。

・Liberty for Java

Cloud FoundryベースのJavaビルドバックでLibertyを採用しています。Webアプリ(.war)をデプロイするだけで、Liberty上稼働するサービス提供が可能です。

またアプリサーバーだけでなく、DBや分析サービス、その他様々なBluemix上のサービスと容易に連携が可能であり、必要となるコンポーネントを組み合せてアプリ開発を進めることができます。

・Libertyコンテナ

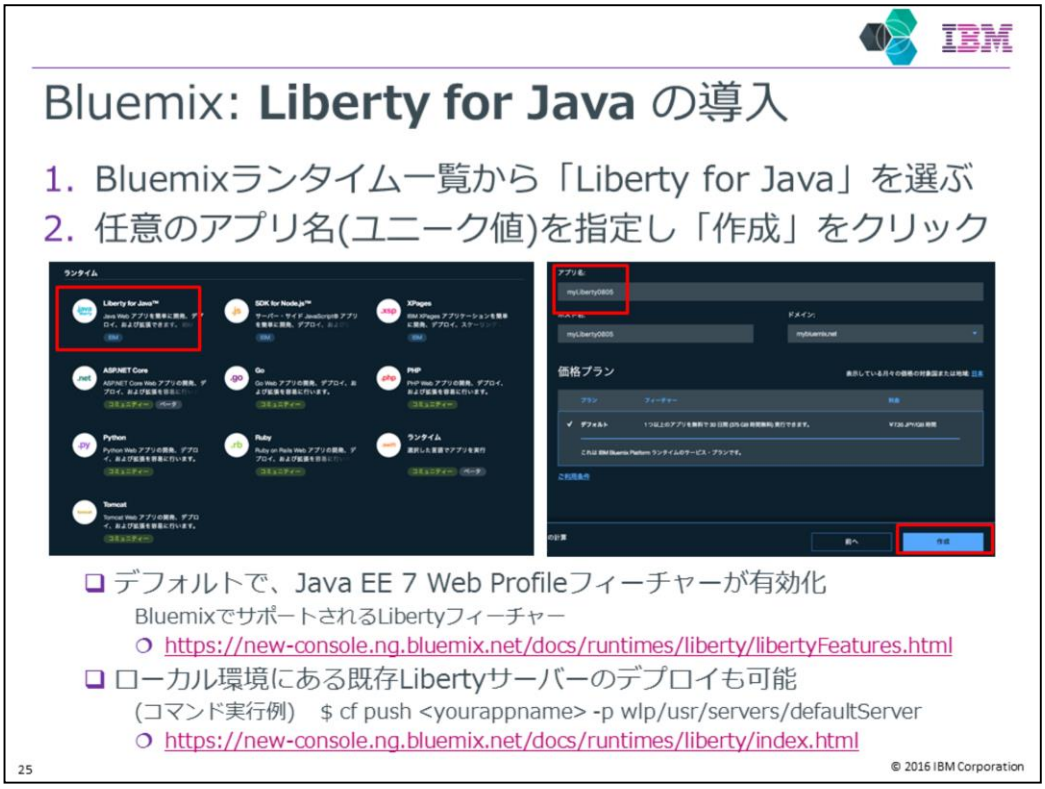
LibertyのDockerイメージを使用して、Bluemix上でDockerコンテナ稼働させることができます。

複数台構成、可用性構成なども柔軟に設定できます。

・WAS for Bluemix

Liberty環境が構成済のVMインスタンスを提供します。SSHのログインも可能で、OS設定も変更でき柔軟性が高いBluemixサービスです。

また、Libertyだけでなく、traditional WASも選択可能であるため、要件にあったランタイムの検証にも活用できます。



【Liberty for Java】

デフォルトでは、Java EE 7 Web Profileフィーチャーが有効化されたランタイム環境が生成されます。

Java EE 7 Full Platformのフィーチャーも使用可能です。使用可能なフィーチャーは以下を参照ください。

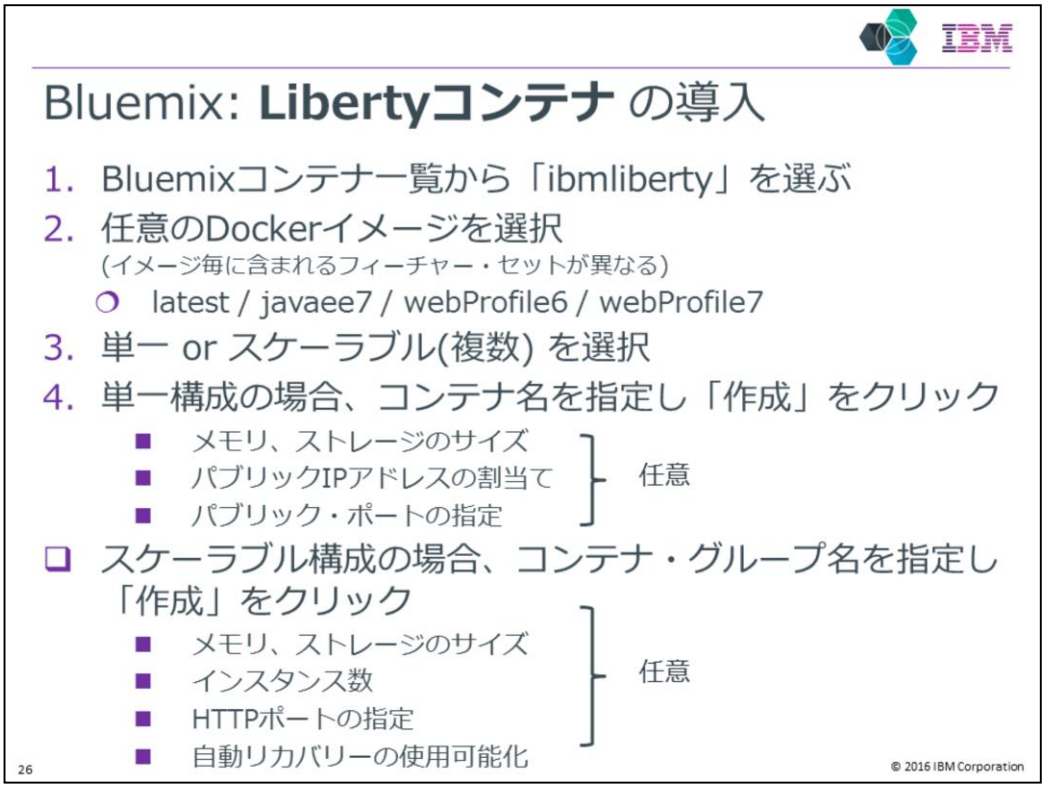
https://new-console.ng.bluemix.net/docs/runtimes/liberty/libertyFeatures.html

BluemixのLiberty for Javaランタイムは、アプリをデプロイするだけで、すぐに外部に向けてサービス提供が可能です。

更に、サーバー構成ファイルなどローカルの開発PCで用意したLiberty環境をBluemix上にデプロイすることも可能です。

詳しくは以下を参照ください。

https://new-console.ng.bluemix.net/docs/runtimes/liberty/index.html



【Libertyコンテナ】

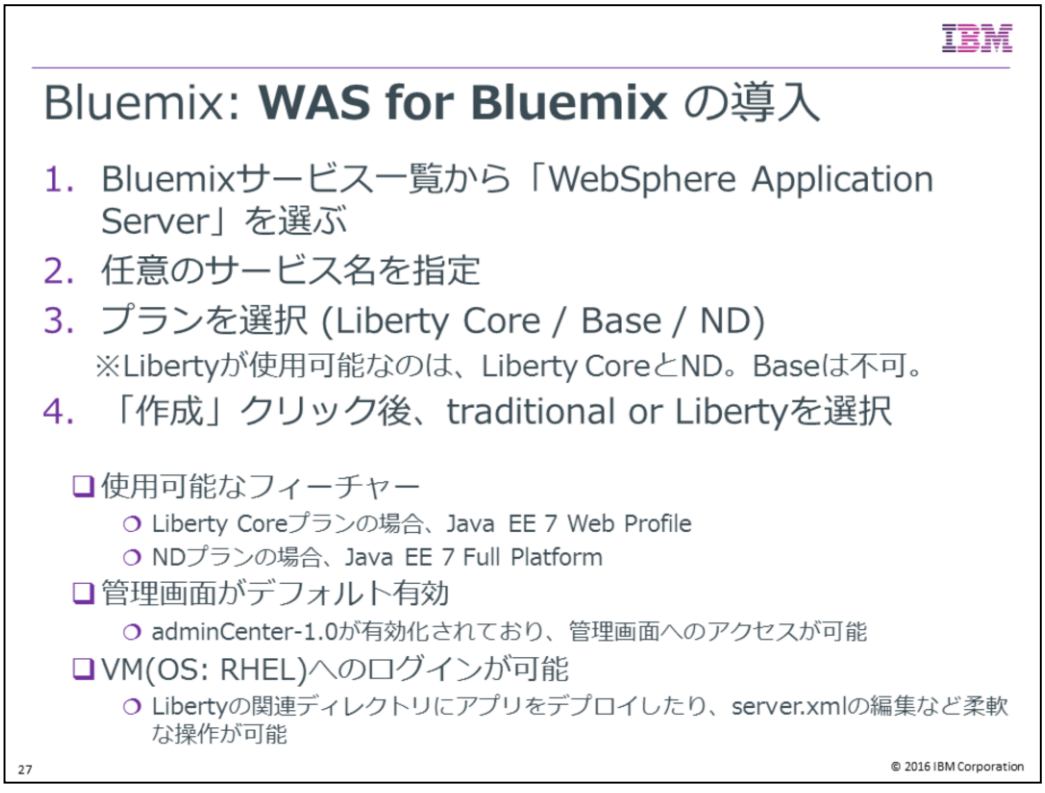
IBMは、DockerHubにLibertyのDockerイメージを公開しています。これはどなたでも開発用途に無償で利用可能なイメージです。

Bluemix上のLibertyコンテナは、公開しているLibertyのDockerイメージを使用して、Dockerコンテナを構成します。

Bluemix上ですぐに本番利用して頂くことが可能です。

ユーザーは、Libertyイメージ一覧から、任意のフィーチャーセットのLibertyを選択することでコンテナ生成が可能です。

この他、スケーラブル(複数)構成やそれに伴うインスタンス数の指定なども行えます。



【WAS for Bluemix】

プランとして、Liberty Core/Base/NDから選択可能です。
ただし、Libertyが使用可能なのは、Liberty CoreとNDプランのみとなりますので注意ください。

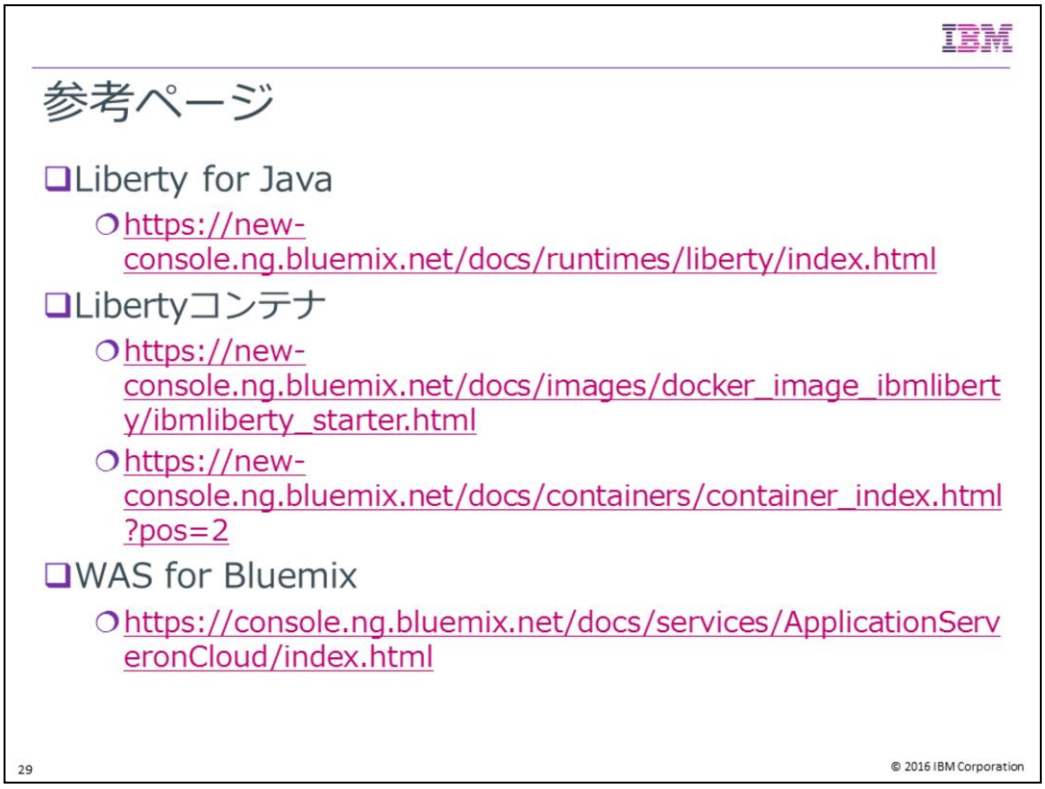
Liberty CoreとNDの選択基準のひとつは、利用予定のJava EEフィーチャーです。
Java EE 7 Web Profileのみで良い場合は、Liberty Coreプランを。
Java EE 7 Full Platformが必要な場合は、NDプランを選択します。

※NDプランに関しては、LibertyのVMに加えてIHS用のVMを提供します。複数台のLibertyランタイムでクラスター構成を組む場合には、NDプランを選択します。

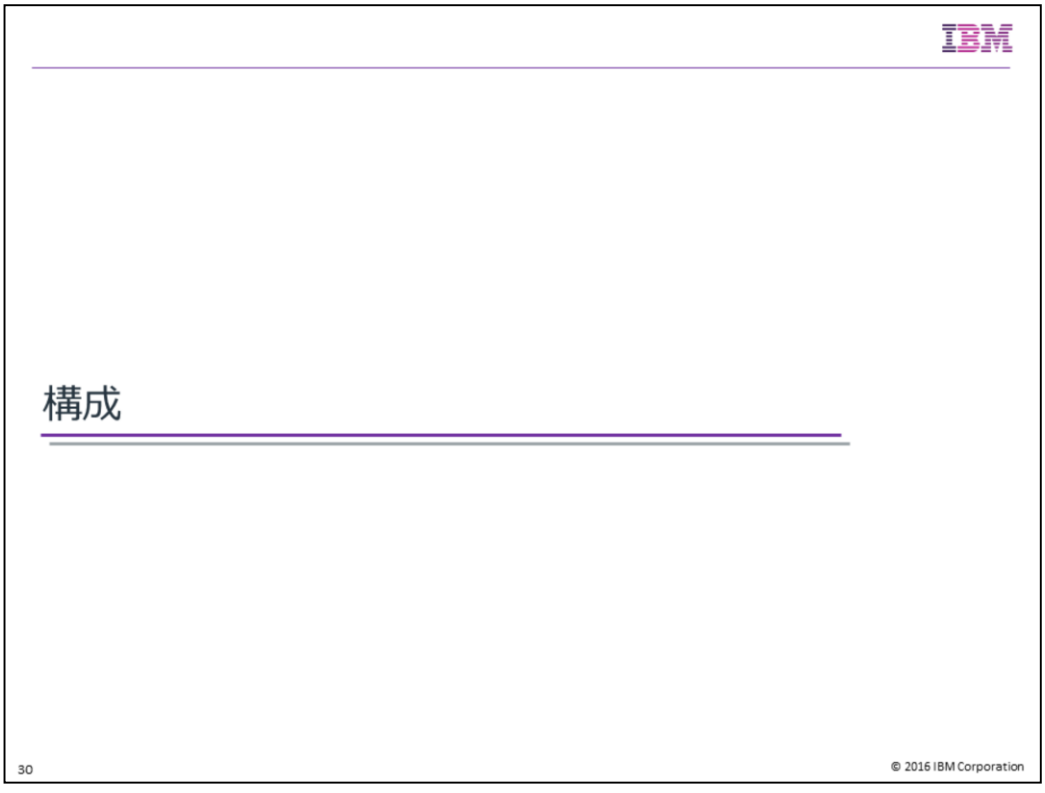
Bluemix上の各Libertyのバージョンをプロビジョニング前に確認する方法です。

各ページは、最新情報を得るために、ブラウザのコンテンツ言語を英語にして閲覧するようにしてください。

(注意) あくまで、2016/8/5時点の方法であり、かつBluemixの他のランタイムやサービスが共通であるとは限りません。



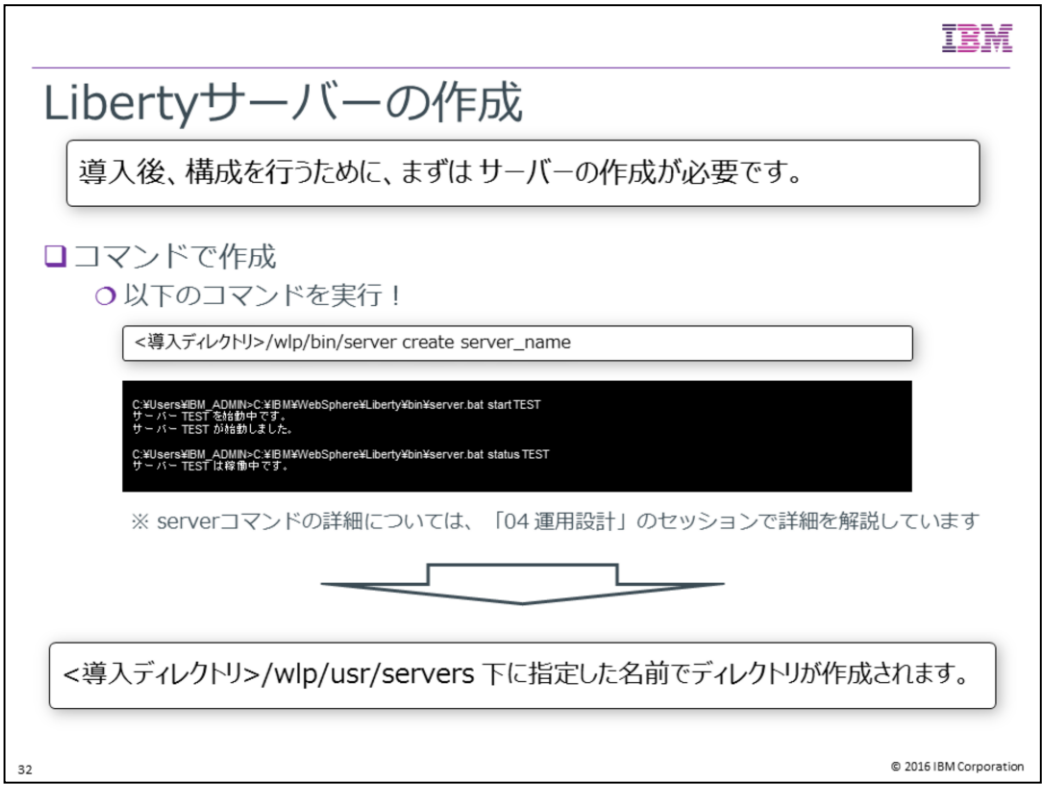
Bluemix上の各Libertyに関しては、スライドのリンクを参照ください。



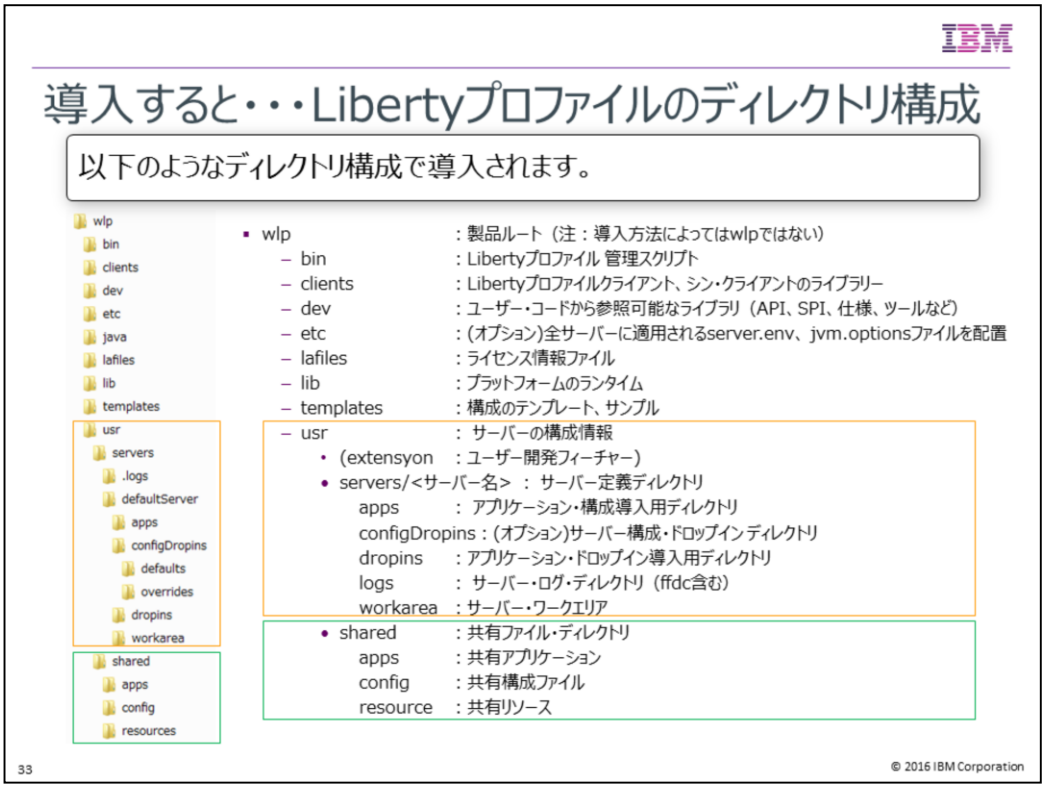
ここからは、どのようにLibertyを構成するか、を解説します

アジェンダ

- 導入
  - ランタイムの導入方法
    - IBM Installation Manager / アーカイブ展開
  - 実働環境への導入作業の流れ
    - ランタイム・Java / Fix / フィーチャー
  - BluemixにおけるLibertyの導入
- 構成
  - 構成要素
    - ディレクトリ構造
    - 使用可能な環境変数
    - 構成ファイル（server.xml他、include可能、などの特徴）
  - 構成 の おまけ機能
  - 「構成」の運用を考える
    - サーバー構成をアーカイブする
    - ディレクトリ構成運用を考える
    - サーバー構成ファイルの運用を考える
  - 構成の流れ と 構成手順
    - 1. スタンドアロン構成
    - 2. シンプル・クラスター構成
    - 3. 高可用性構成

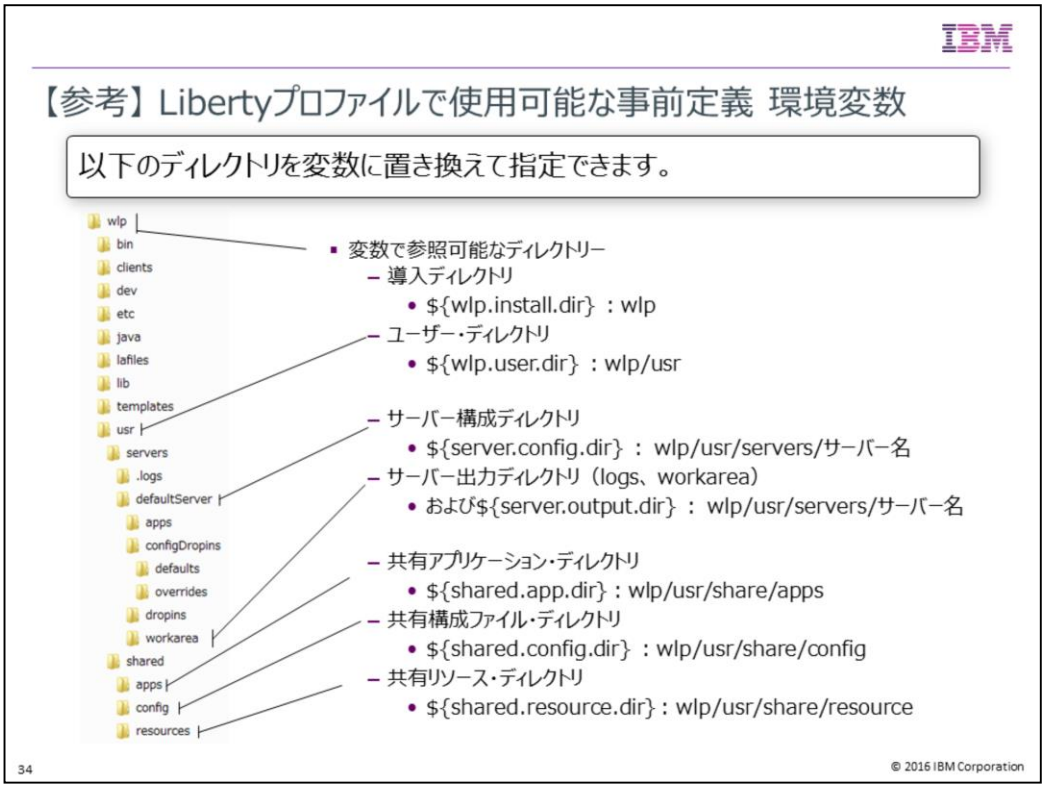


導入が終わったら、構成に先立ってまずは構成の対象となるサーバーを作成する必要があります。

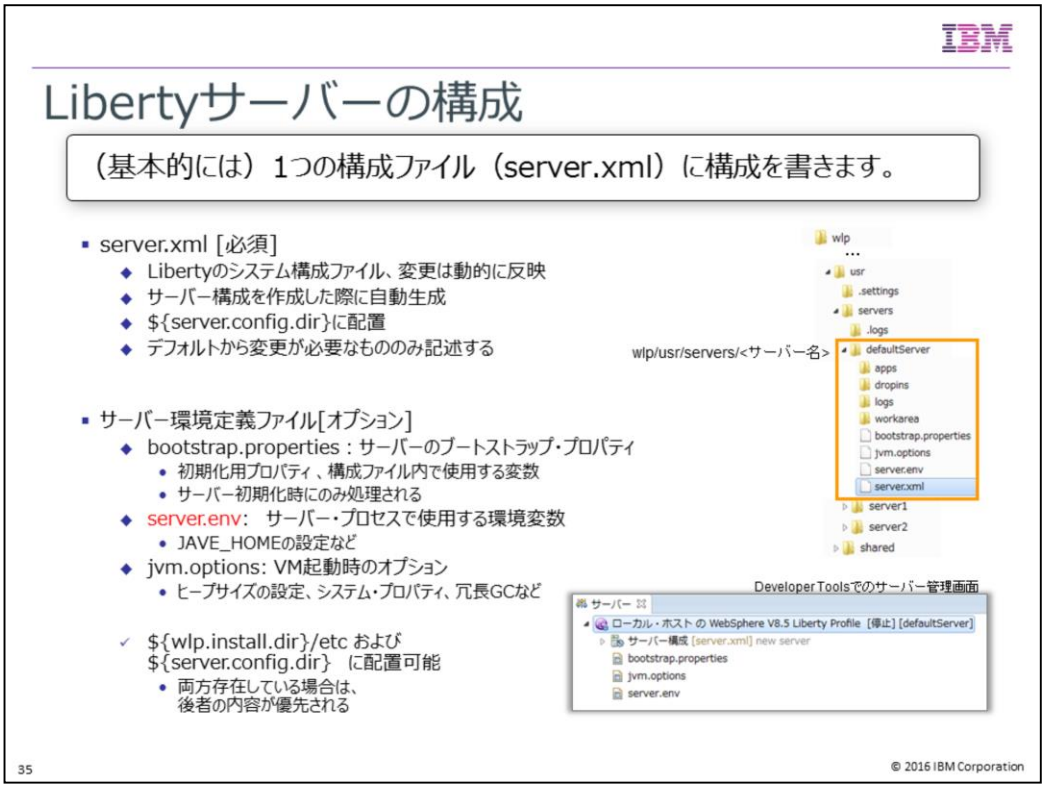


導入したあとのディレクトリ構成は、上記のようになっています。
大きくは、usr以下に、個別のサーバー設定やアプリケーションが配置されます。
usr下のserver以下には、作成したサーバーの構成情報が格納されます。
usr下のshared以下には、サーバー内/サーバー間で共有されるアプリケーションやライブラリを配置します。

Liberty プロファイル: ディレクトリーのロケーションおよびプロパティー
https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_dirs.html



このディレクトリ構成において、主要なディレクトリについては、変数で参照することが出来るようになっています。



それでは、Libertyプロファイルを構成する構成ファイルについてみていきましょう。まず唯一必須となる構成ファイルが server.xmlです。シンプルな構成であればこのserver.xmlのみで構成が完了します。サーバー構成ディレクトリ直下に作成されます。

さらにその他として、追加で作成・利用可能なファイルが３つあります。1つ目が、サーバーの初期化時に読み込まれるbootstrap.propertiesです。構成ファイル内で使用する変数であったり、トレース・ファイル定義だったりを指定することが可能です。サーバー・プロセスが読み込む環境変数を定義する server.envファイルや、JVM起動時オプションを設定することができる jvm.optionsファイルがあります。ヒープサイズの設定やverbose:gcの設定はここで実施します。これらのファイルはサーバー起動時に読み込まれるため動的変更は行われません。

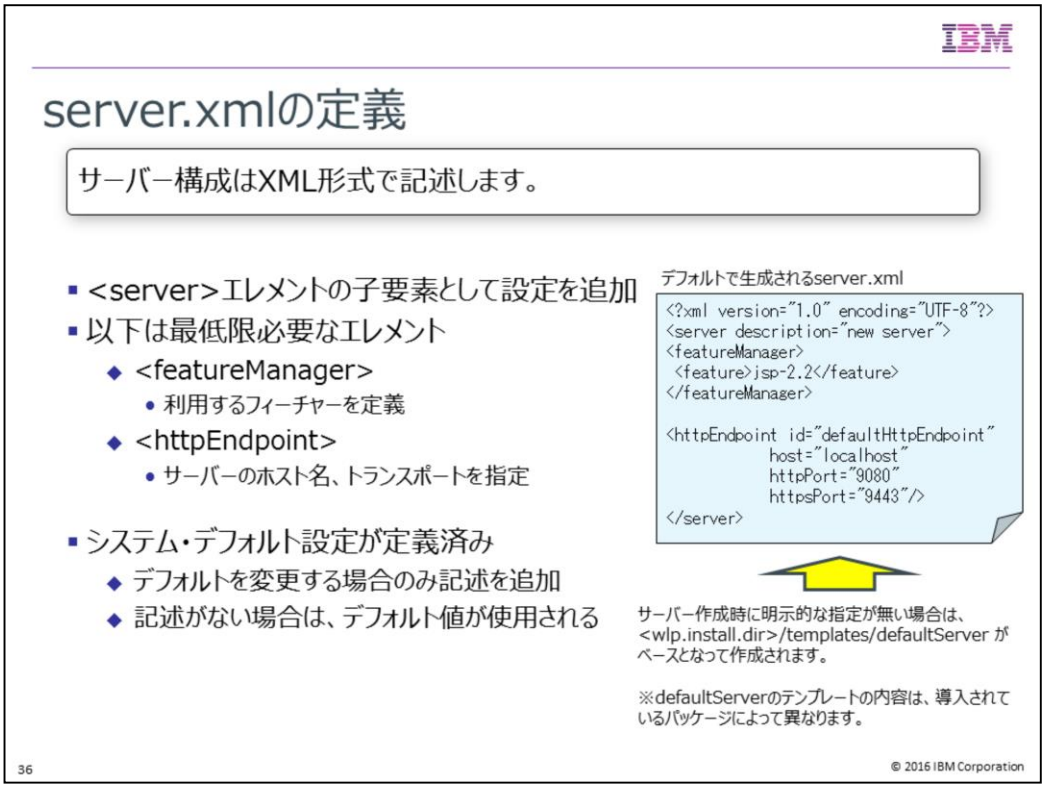
さらに詳細な内容については以下のKnowledge Centerを参照ください。

Specifying Liberty profile bootstrap properties

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_inst_bootstrap.html

Customizing the Liberty profile environment

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_admin_customvars.html



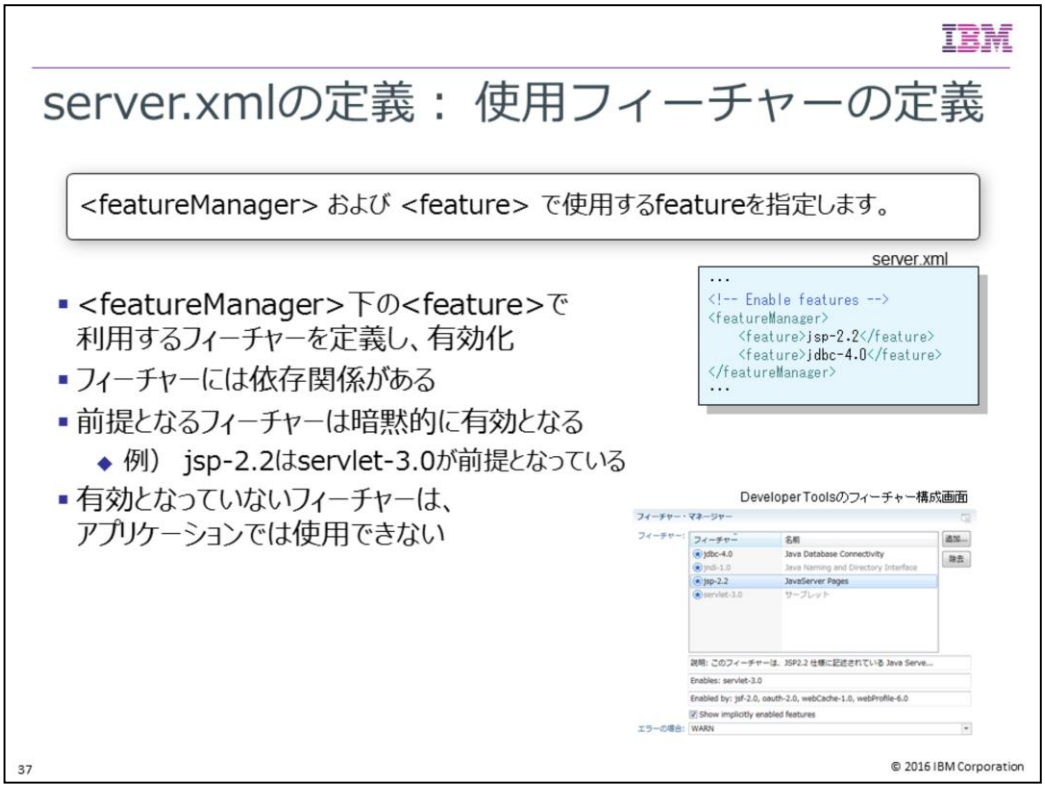
server.xmlについて、より詳しく見ていきます。

右にあるXMLファイルが、デフォルトで作成される server.xmlファイルです（※）。この構成ファイルでJSPとServlet稼動するWebコンテナーについて定義が完了しています。従来のWASの構成ファイルをご存知の方はそのシンプルさに驚くことと思います。

簡単にXMLの要素を見ていきます。<server>エレメントの下に<featureMaanger>エレメントがありこの中でJSPが有効化されています。さらに<httpEndpoint>エレメントがありWebコンテナーがListenするホスト名とポートが指定されています。

それ以外は全てシステムのデフォルト値が暗黙的に定義されています。デフォルトを変更する場合のみ明示的にserver.xmlに記載します。記載がない場合は、デフォルト値が有効になるため、構成ファイルをシンプルに保つことができています。

※ server.xmlのデフォルトの設定は、導入したパッケージによって差異があります。



<featureManager>の下には、<feature>エレメントを追加して構成していきます。

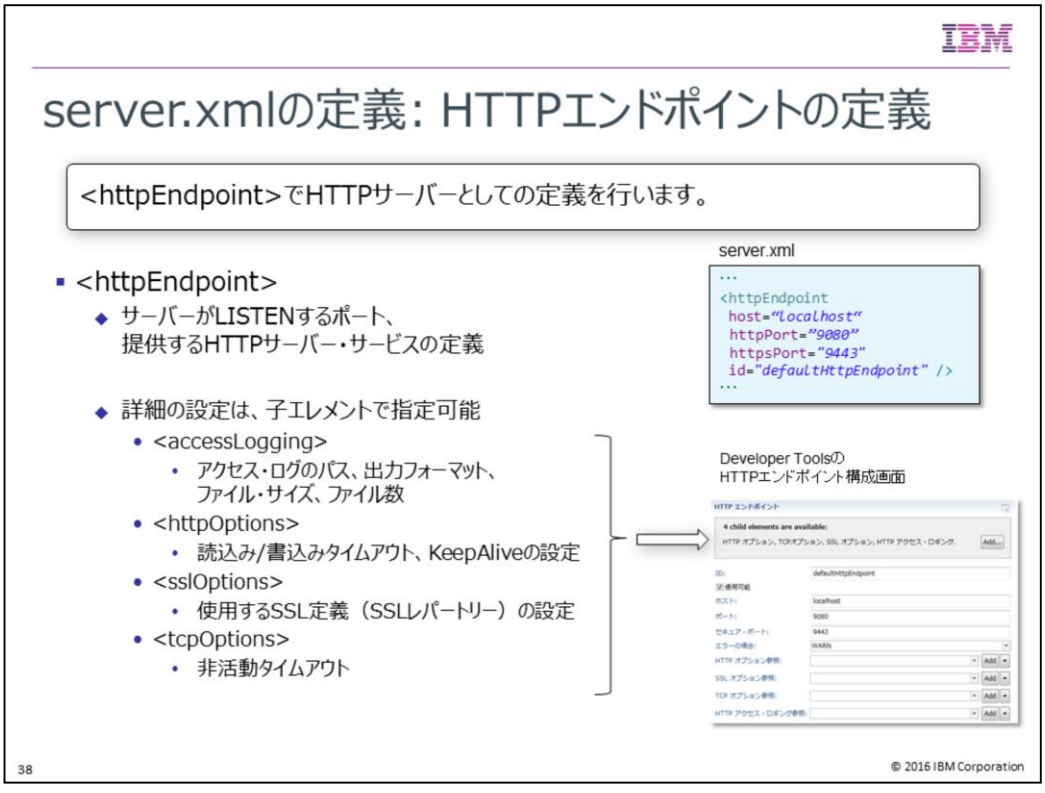
各<feature>には依存関係が定義されていて、前提となるフィーチャーは暗黙的に有効にされます。たとえば、jsp-2.2に関してはservlet-3.0が前提となっているため、jsp-2.2を有効化すると、暗黙的にservlet-3.0が有効化されます。

server.xmlでは、明示的に有効化したフィーチャーしか記述されませんが、WebSphere Developer Toolsのフィーチャー構成画面（後述）では、暗黙的に有効とされているフィーチャーに関しても表示することが可能です。

フィーチャーの一覧と利用可能なエディションの対応については、以下を参照ください。

Liberty features

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_feat.html

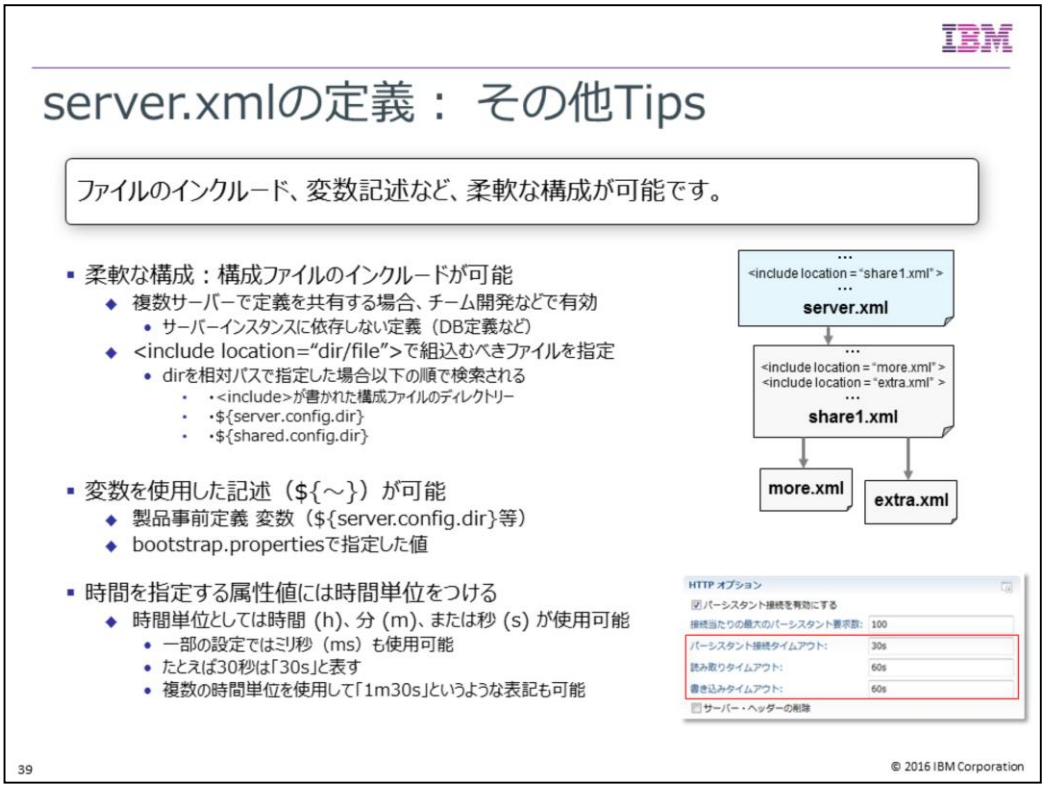


次に<httpEndpoint>です。サーバーのWebコンテナーがListenするポートを指定します。

デフォルトではホスト名とポートだけですが、その他の詳細な設定は、子エレメントで指定していきます。

たとえば、Webコンテナーのアクセスログのフォーマットやサイズなどは、<accessLogging>エレメントで指定していきます。他にもHTTPやTCPのタイムアウトなども子エレメントの中で指定していきます。

特にホスト名やポートの明示指定がないと、暗黙的なデフォルト値で起動します。同一ノード内で複数プロセスを起動する場合などは同一のデフォルト値で起動した結果、ポートがバッティングして正常に起動しない、といった事象につばがるケースがありますので、ホスト名、ポートなどの固有情報を明示指定を行うようにしてください。



さらにserver.xmlは、構成ファイルの一部を別XMLファイルに外出しにすることが可能です。

たとえば、データ・ソースへのアクセスするための定義など、サーバー・インスタンスごとに変わらない共通する定義は、別のXMLファイルとして外出しし、<include>エレメントでserver.xmlに取り込むことが可能です。このように共通する構成ファイルをコンポーネント化することで管理の容易化を実現できます。またチーム開発などの場合も、サーバー・インスタンスに依存しない共通構成ファイルを配ることで、簡単に構成の共通化を実現することが可能です。

その他、bootstrap.propertiesで定義した変数を構成ファイルに利用することができます。従来のWebSphereではタイムアウトを指定する場合は単位を確認する必要がありましたが、Libertyではタイムアウト関連パラメータは、hやm、sなど単位を指定します。たとえば3分は3mと定義することができますし180sと定義することもできます。

Using include elements in configuration files

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_includes.html

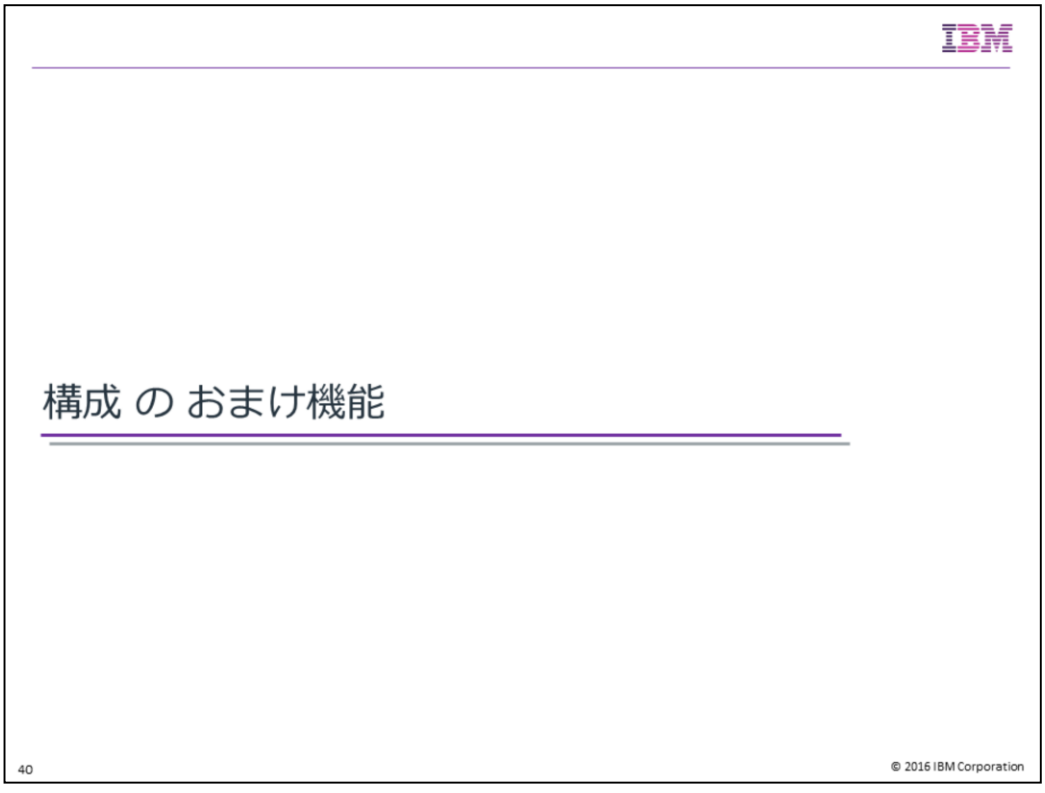
Using variables in configuration files

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_vars.html

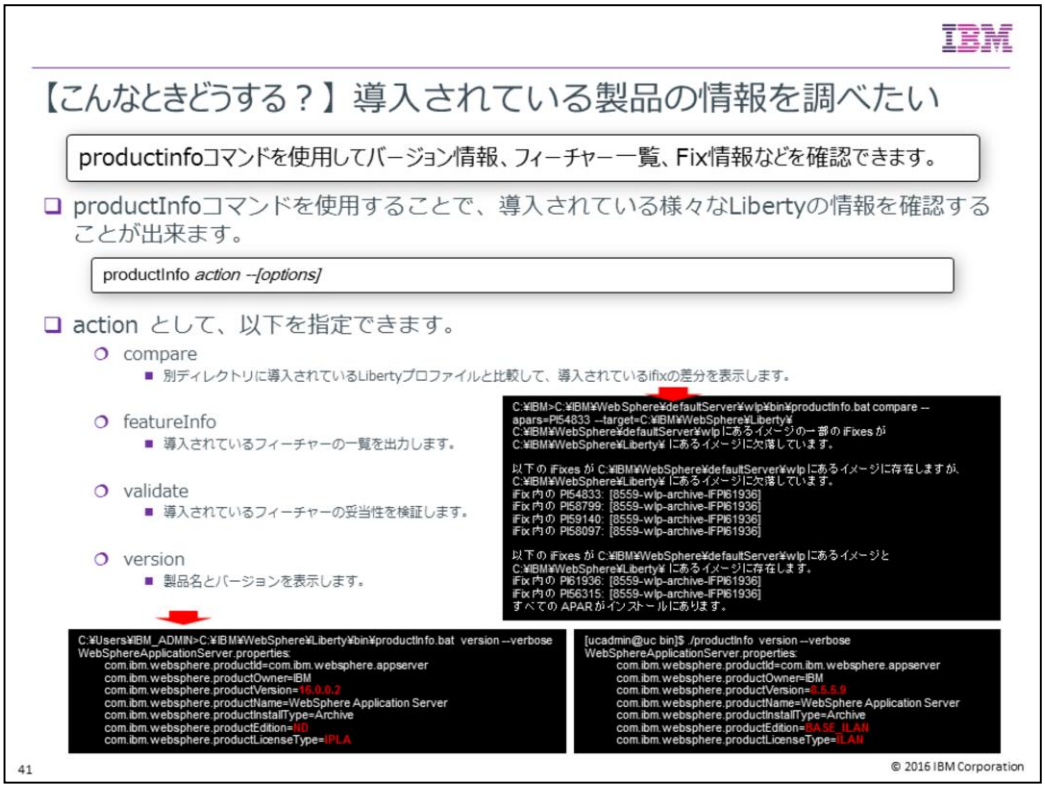
Using Ref tags in configuration files

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.we

bsphere.wlp.nd.multiplatform.doc/ae/twlp_setup_reftags.html



ここからは、構成に関連する便利な機能やTipsを紹介します。

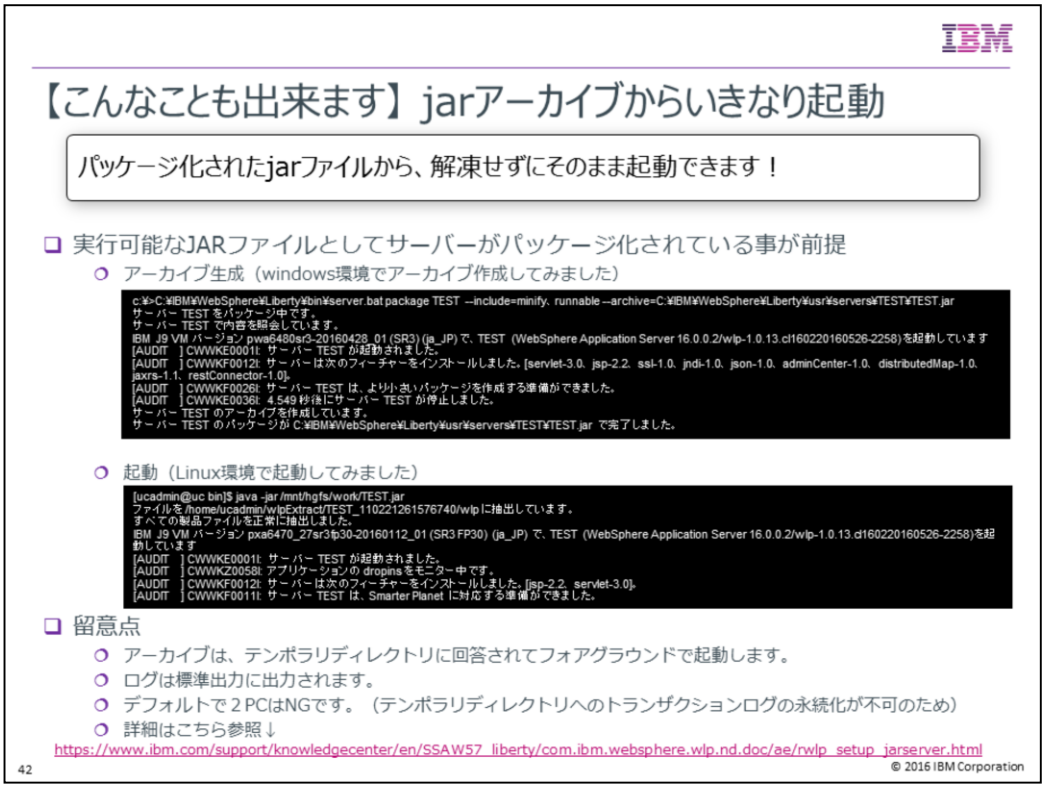


Libertyでは導入する方法もバリエーションがあり、このランタイムはどのように導入したのだっけ？と思うことがしばしばあるかもしれません。

そのような場合は、このproductinfoコマンドが役に立ちます。他にも、導入されている個別fixの内容を調べたり、ライセンスの状況を調べたり、利用するケースが度々あると思われます。詳細については以下を参照ください。

Liberty: productInfo command

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_productinfo.html



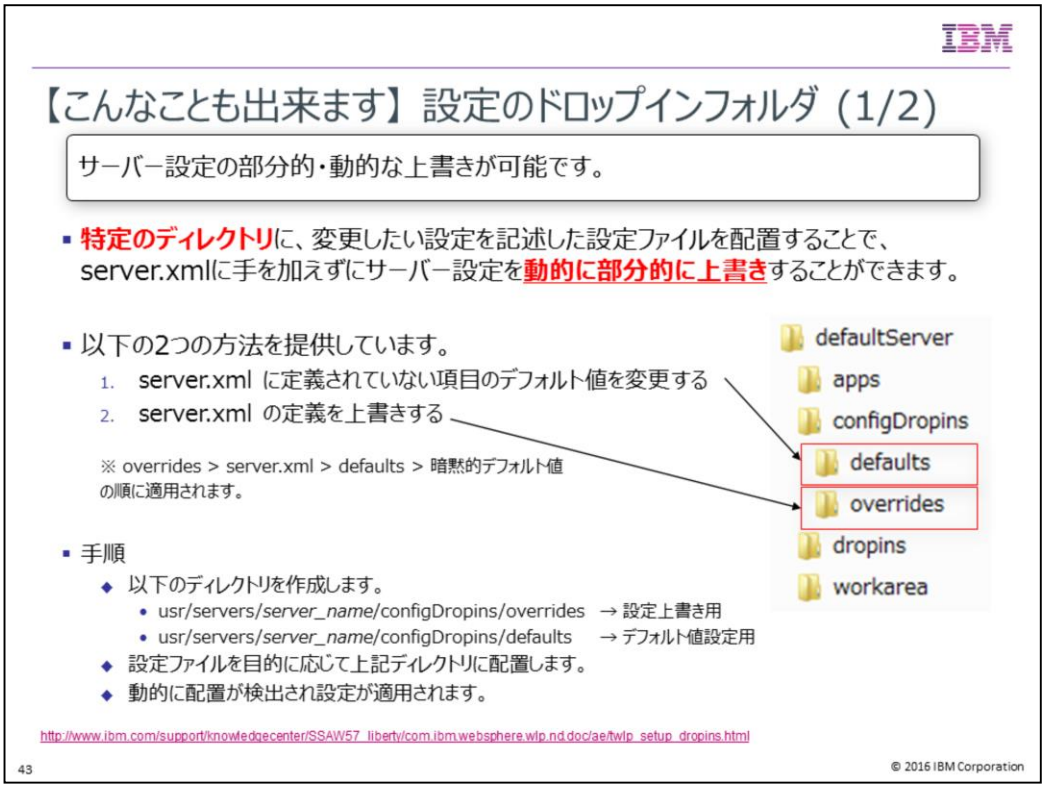
Libertyはその軽量さが売りのひとつですが、この機能はまさにそれを象徴しています。Libertyをアーカイブする際に、起動可能指定を行うと、なんとアーカイブから直接起動できてしまいます。

とはいえあくまでもテンポラリのディレクトリに解凍した上で起動するわけですが、この機構のため、この場合は利用可能な機能に制限が出てきます。

こちらも詳細については以下を参照ください。

Running a Liberty server from a JAR file

https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_setup_jarserver.html



また、server.xmlの変更はモニターされて自動で再ロードされますが、なんとserver.xmlの内容はそのままに、変更したい部分のみを切り出して構成ファイルを作成し、特定のディレクトリに配置することで、一部分のみを動的に上書きする、という事も可能です。

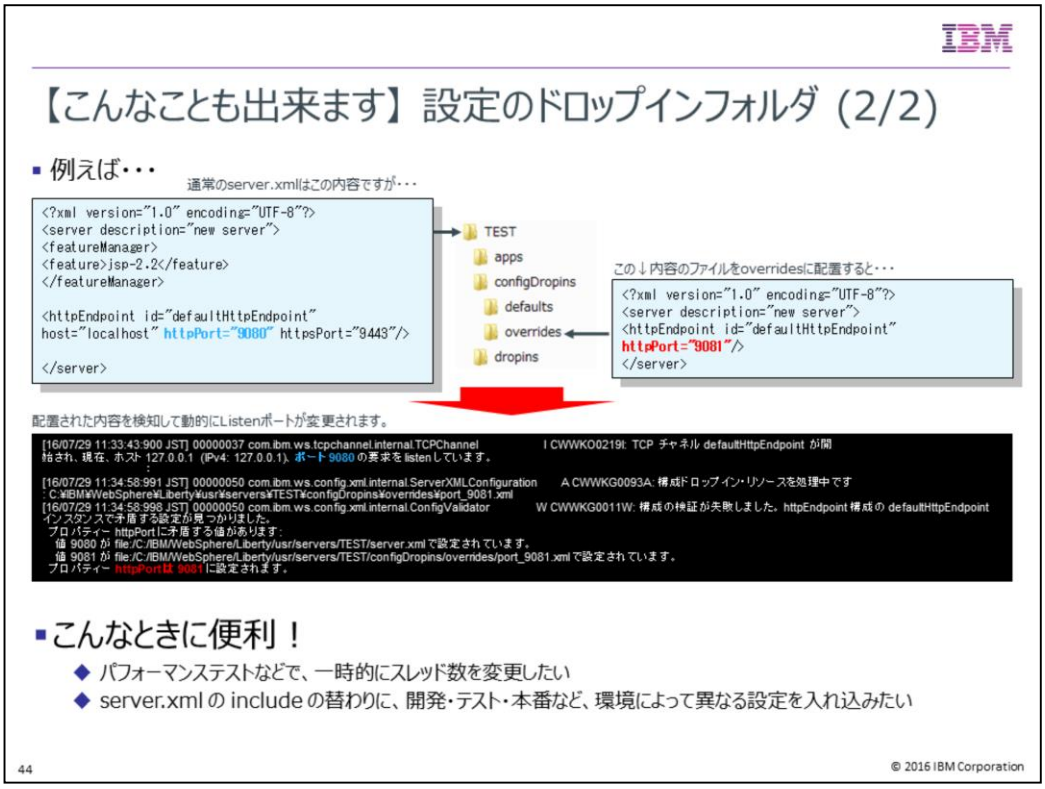
この特定のディレクトリ、というのは、defaultsと、overridesの2種類用意されており、どちらにファイルを配置するかで効き方が変わってきます。

overridesに配置した場合は、単純にserver.xmlの内容を上書きします。

defaultsに配置した場合は、server.xmlに記述されていない設定のデフォルト値を変更することになります。

server.xmlに設定されていない設定項目は、暗黙的にデフォルト値で動作することになりますが、これらの優先順位は以下のようになります。

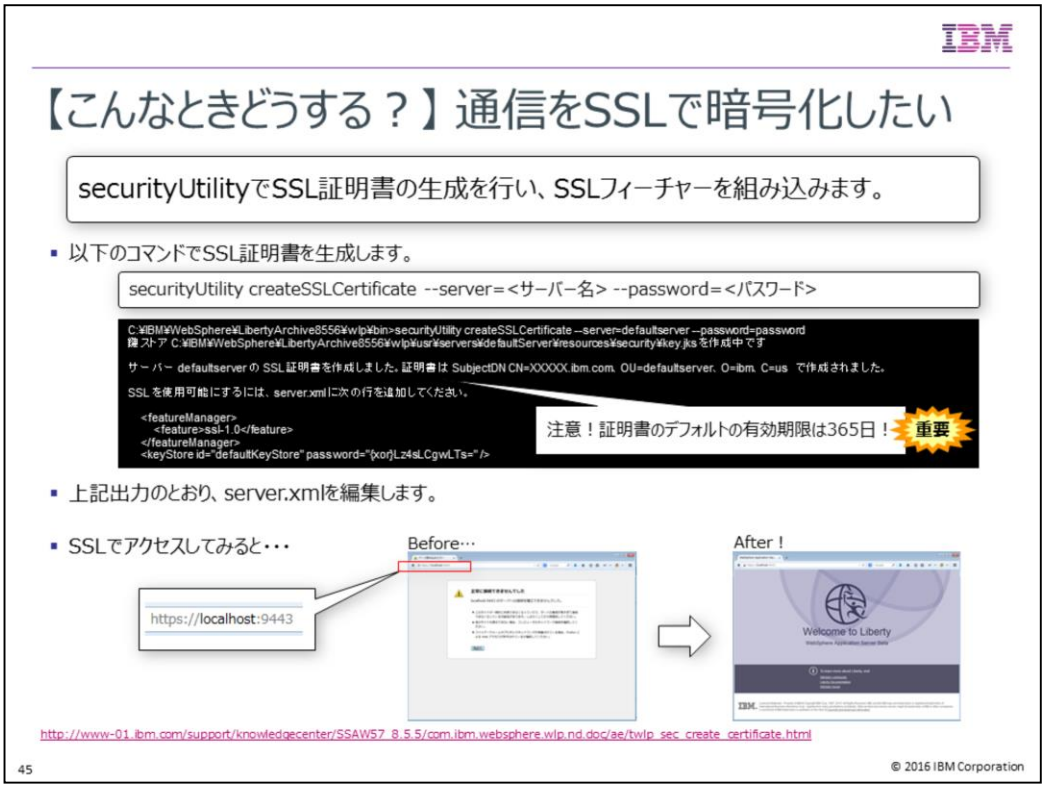
overrides > server.xml > defaults > 暗黙的デフォルト値



上記は実行例です。

この機能は、例えばパフォーマンステスト時に一時的なパラメータを変更してチューニングの効果を確認したいt、という場合や、開発・テスト・本番など、環境によって異なる設定を入れ込みたい場合に便利です。

特にパフォーマンステスト実施時には、テストケースとserver.xmlファイルを関連付けることで、このケースはどういうパラメータで実施したか、というエビデンスにもなりますので非常に便利です。



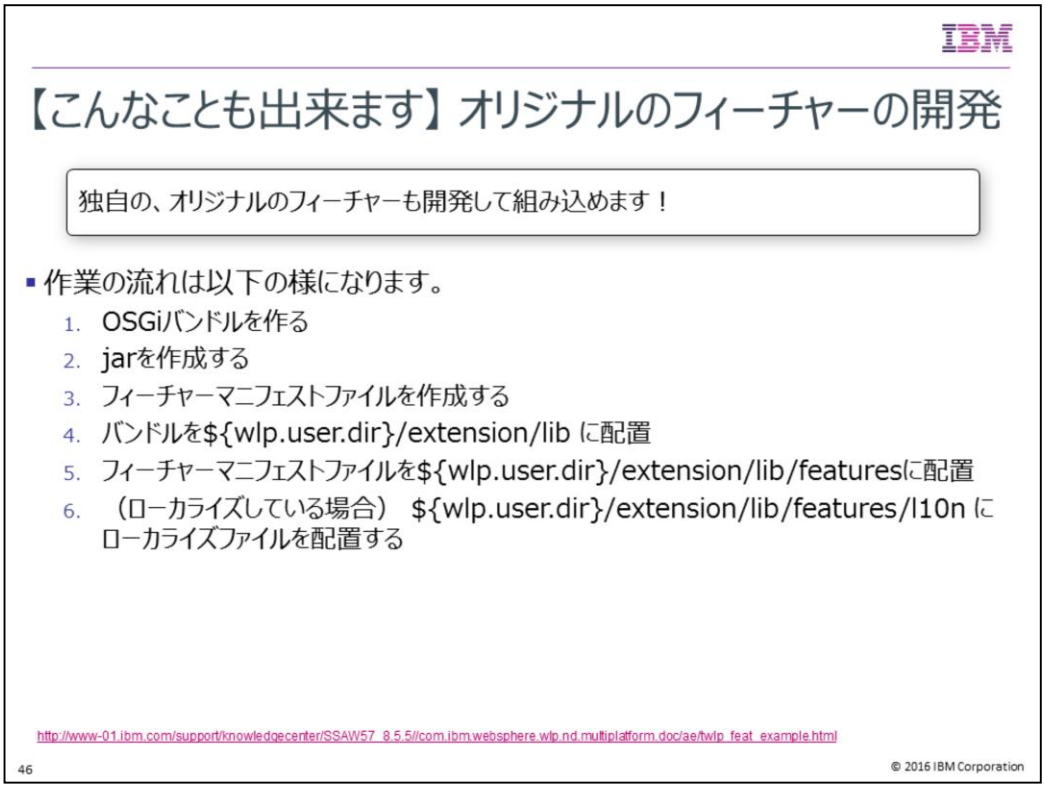
また、Libertyサーバーとクライアント間の通信を、HTTPSにしたい、という場合は上記の手順をとる事で簡単に構成することが可能です。

ただし、詳細なSSLオプションを設定する必要がある場合は、以下を参照してください。

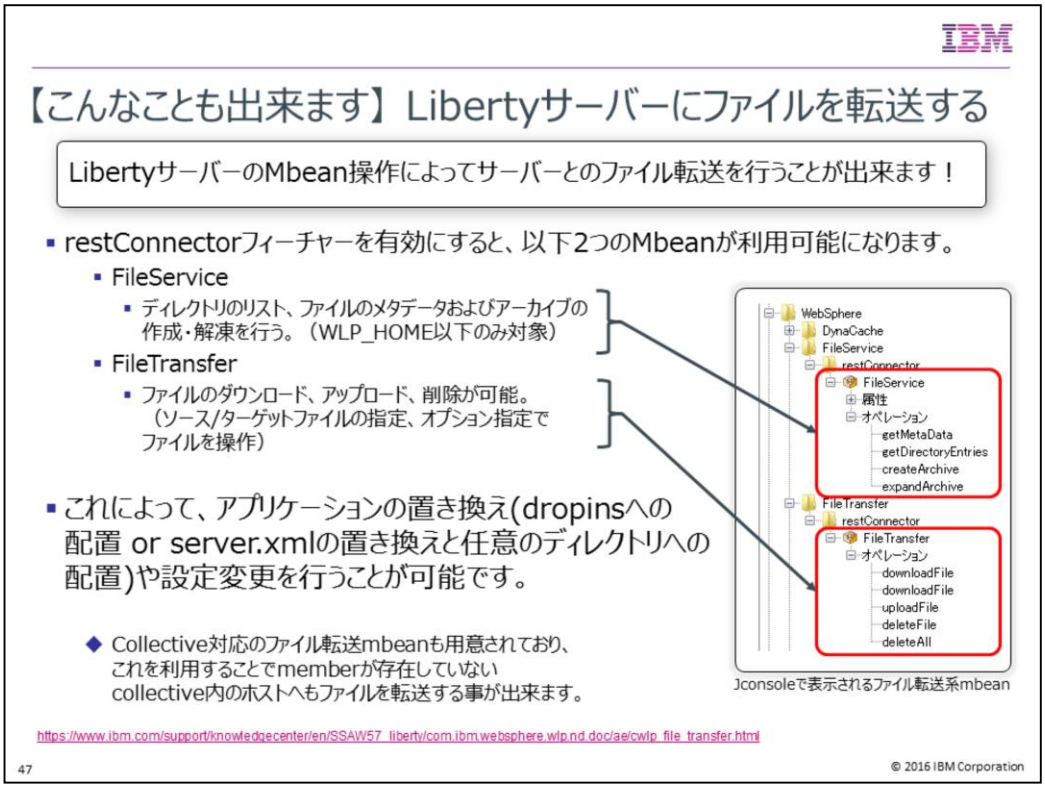
ここでひとつ大事なポイントがあります。デフォルトで作成した証明書は、有効期限は365日、1年になっています。きちんと認識して運用するか、長めに設定しておくか、留意してください。（尤も、本番運用時には自己署名証明書ではなく厳密な管理を行うことと思いますが・・・・）

Liberty プロファイル: SSL 構成の属性

https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_ssl.html



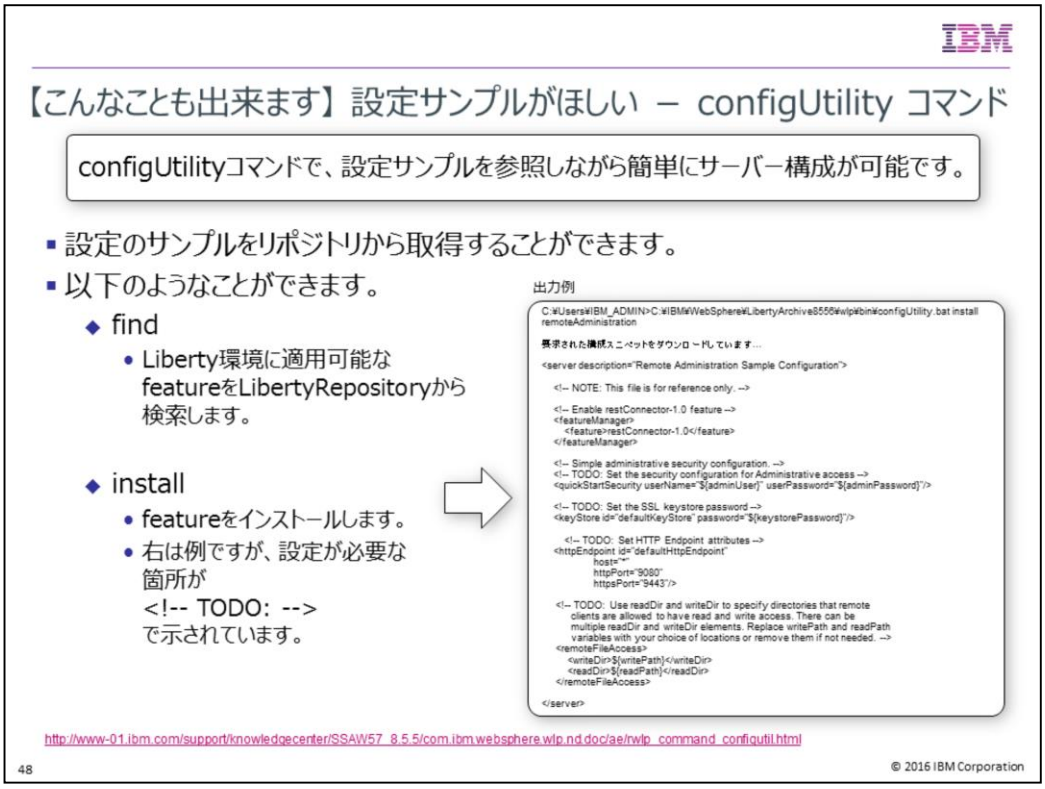
さらに、Libertyではオリジナルのフィーチャーも作成し、組み込むことが可能です。



Libertyには、ファイルの送受信を行う機能もあります。restConectorフィーチャーを利用することで、ファイル操作関連のMbeanと、ファイル転送関連のMbeanが有効になります。

これによって、リモートのLibertyに構成ファイルを転送してそのまま構成変更してしまったり、することが出来ます。

さらに、Collective対応のmbeanも用意されていますので、この機能を利用した環境構築も不可能ではありません。（この方法は運用のセッションで紹介しています。）

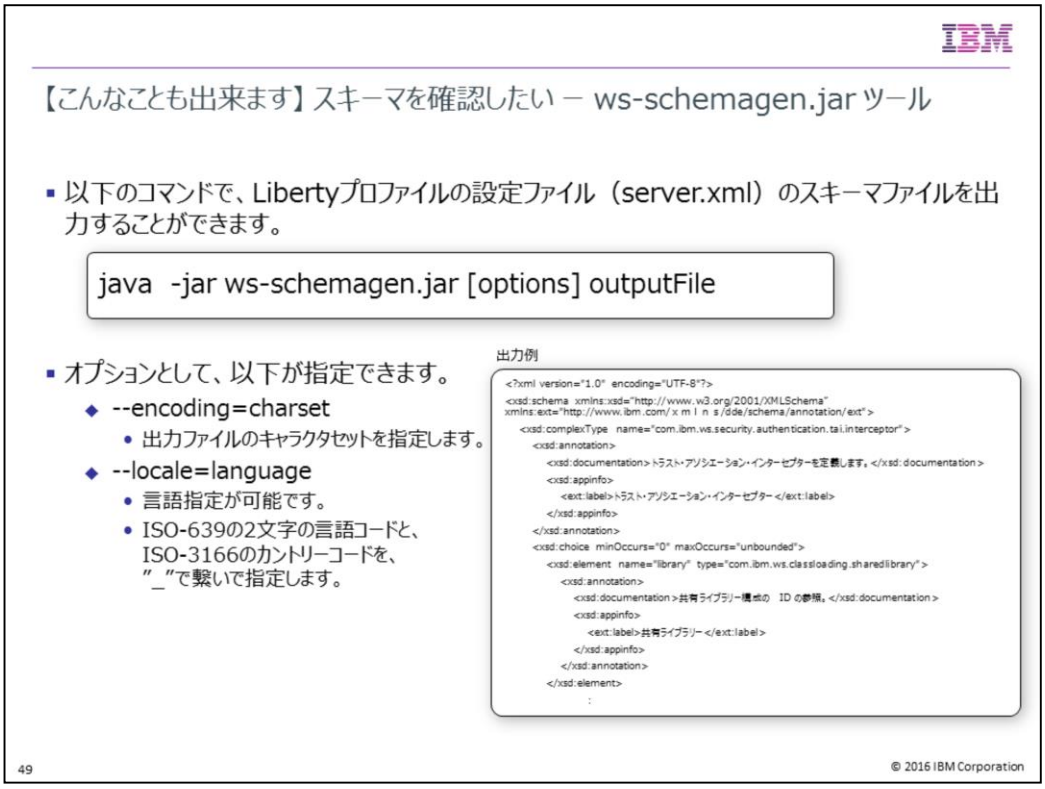


便利情報が続きますが、server.xmlの設定の再に、設定サンプル/テンプレートをダウンロードして参照することができます。

configUtilityというコマンドを利用して、installすると、サンップルが出力されますので、これをベースにして、<!-- TODO: -->で示された部分を編集することで設定を行うことができる便利コマンドです。

Liberty profile: configUtility command

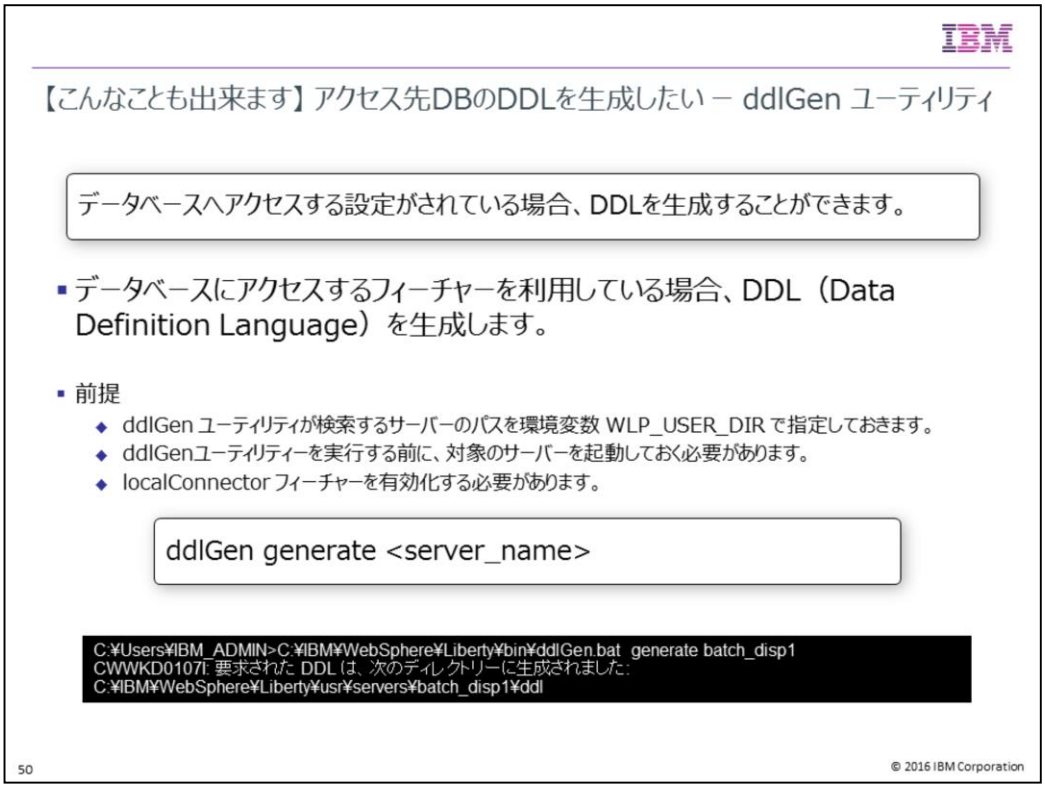
https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_configutil.html



Libertyプロファイルはいろいろな便利ツールが用意されていますが、さらにはserver.xmlファイルのスキーマファイルまで出力するコマンドが用意されています。

Liberty プロファイル: コマンド・プロンプトから Liberty プロファイル構成スキーマを生成

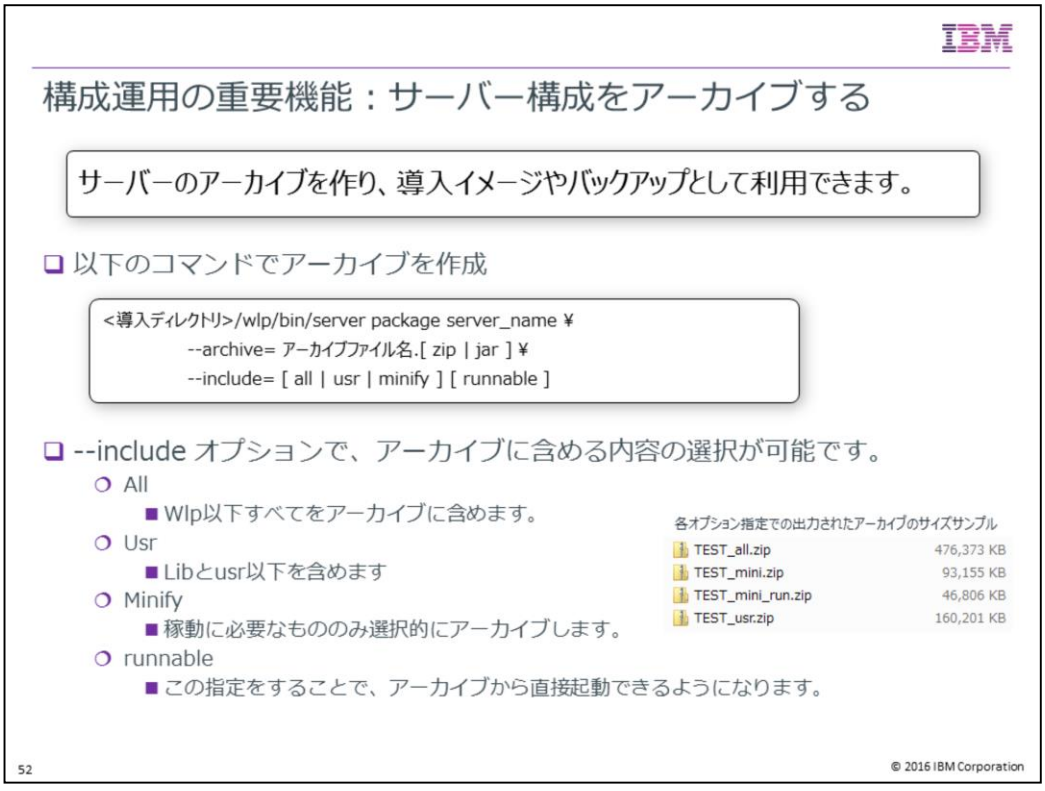
https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/rwlp_command_configutil.html



まだまだ続きます。

DBアクセスを行うフィーチャーを利用している場合には、アクセス先DBのDDLを生成するというユーティリティーまで用意されています。

「構成」の運用を考える



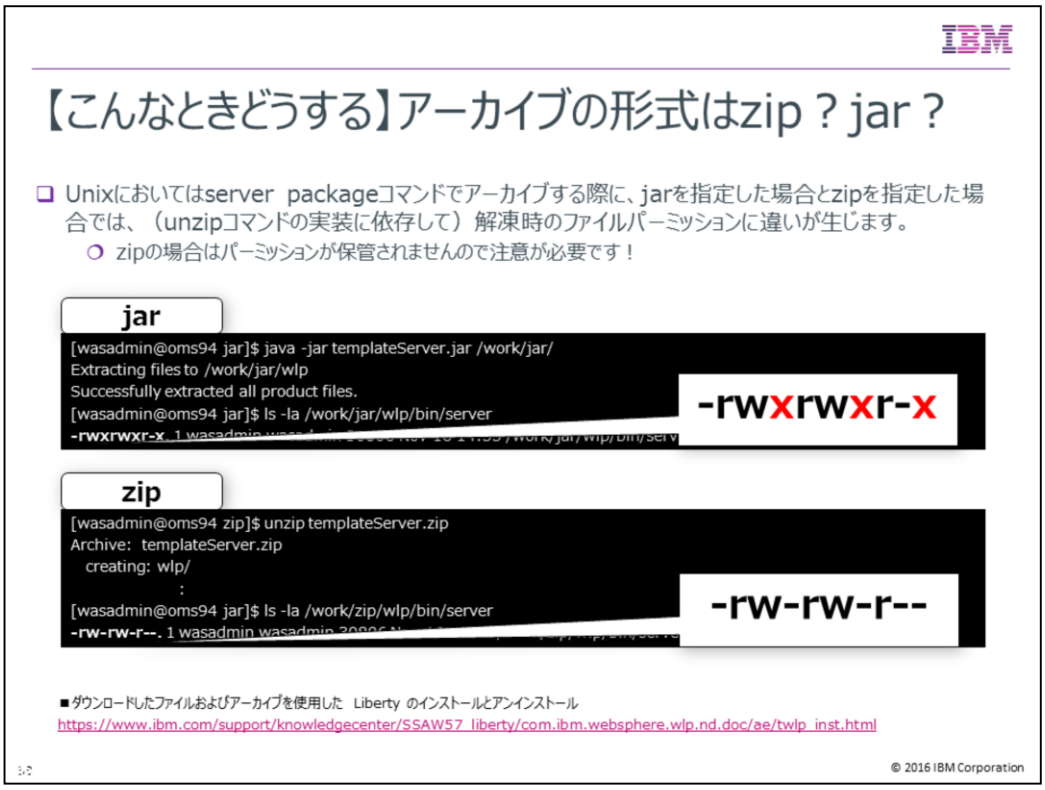
Libertyの構成に関する運用を考えるにあたっては、サーバーのパッケージ機能が非常に重要です。

作成したアーカイブを新規サーバーのテンプレートとしたり、バックアップとしたりすることで、運用を効率的に行うことが出来るようになります。

詳細については以下を参照ください。

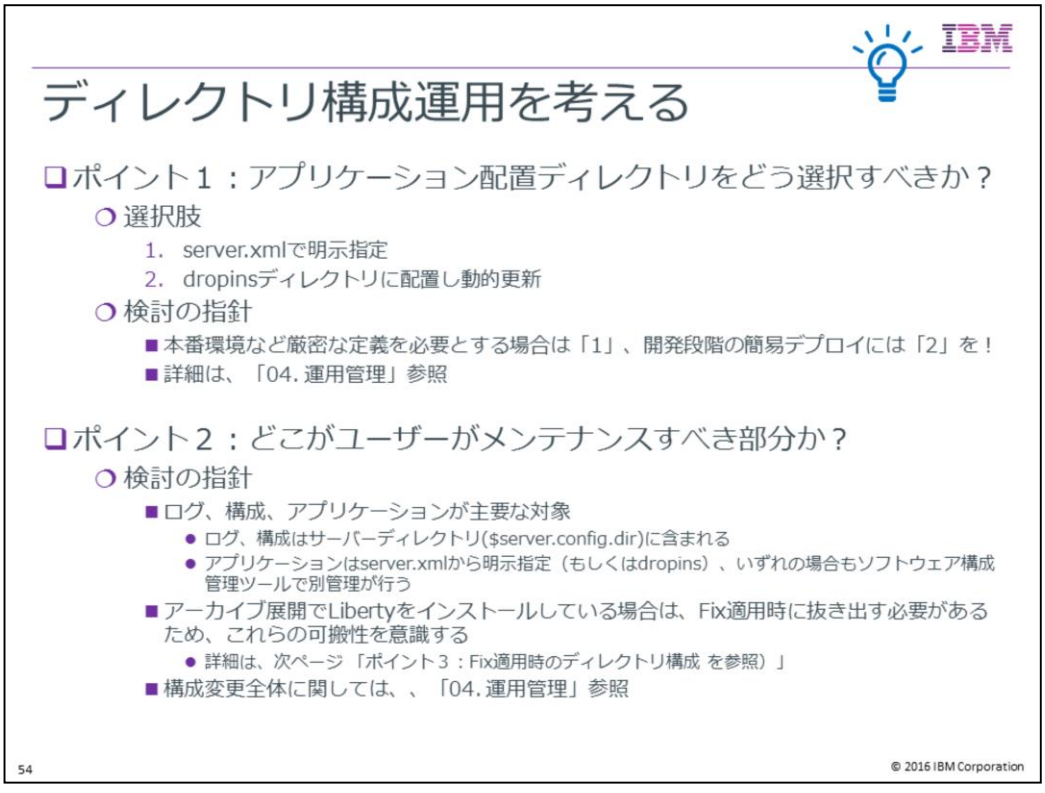
Packaging a Liberty server from the command line

https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.doc/ae/twlp_setup_package_server.html



アーカイブを扱う際のTipsです。

Jarファイルを解凍した場合は、ファイルのパーミッションはそのまま保管されますが、zipの場合は保管されませんので、解凍後もう一手間必要になります。ご注意ください。



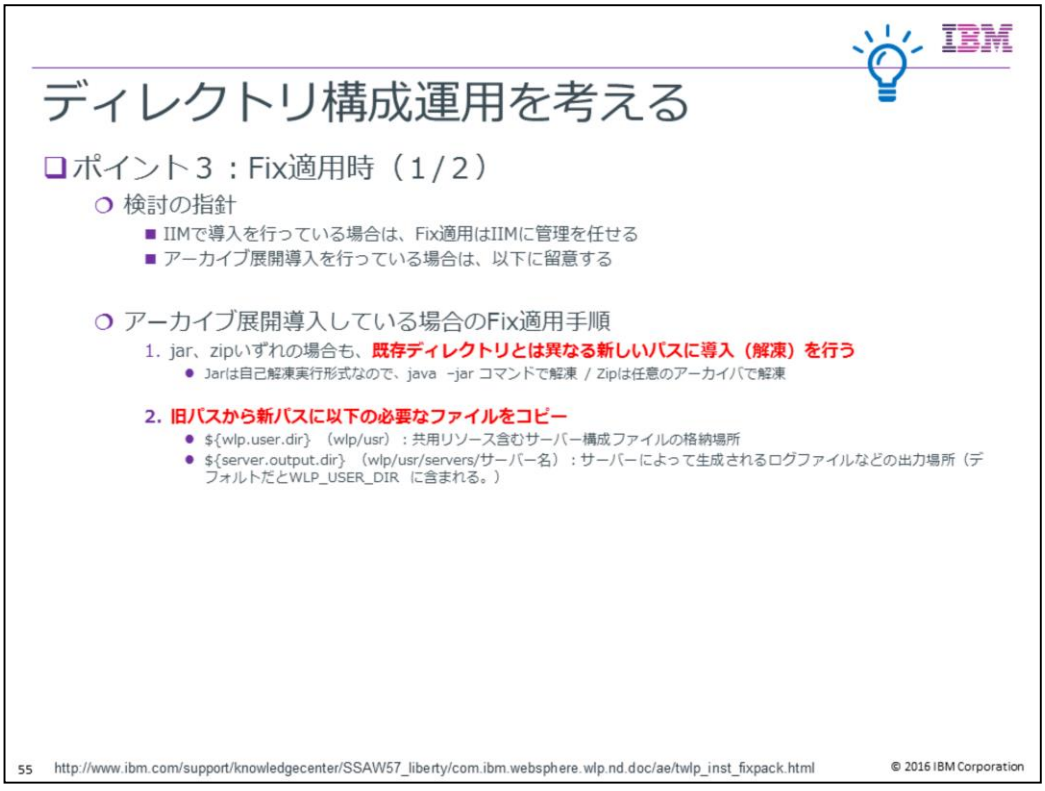
Libertyでは、ディレクトリ構成をどのように監視していくかという運用も検討すべき事項の一つになります。幾つかポイントを挙げます。

1つ目のポイントとしては、アプリケーションを配置するディレクトリをどのように管理するか、という点です。Dropinに動的に配置して動的にロードする、という方法は、主に開発時などに開発環境で非常に便利な使い方になるかと思います。

一方で、明示的に配置場所を指定して、自動的なロードを行わないといった選択肢もあります。実働環境などでは、不用意にアプリケーションの置き換えが行われない用にこちらの方法を採った方が良いでしょう。

2つ目のポイントは、Libertyを構成するディレクトリのうち、どこがメンテナンスの対象になる部分か、という点です。ログファイルの書き出し先やアプリケーションの配置先などを明確にしておく必要があります。

特にアーカイブ導入を行っているケースで、Fix適用時などに問題になります。この点は、次ページでも詳細を解説します。

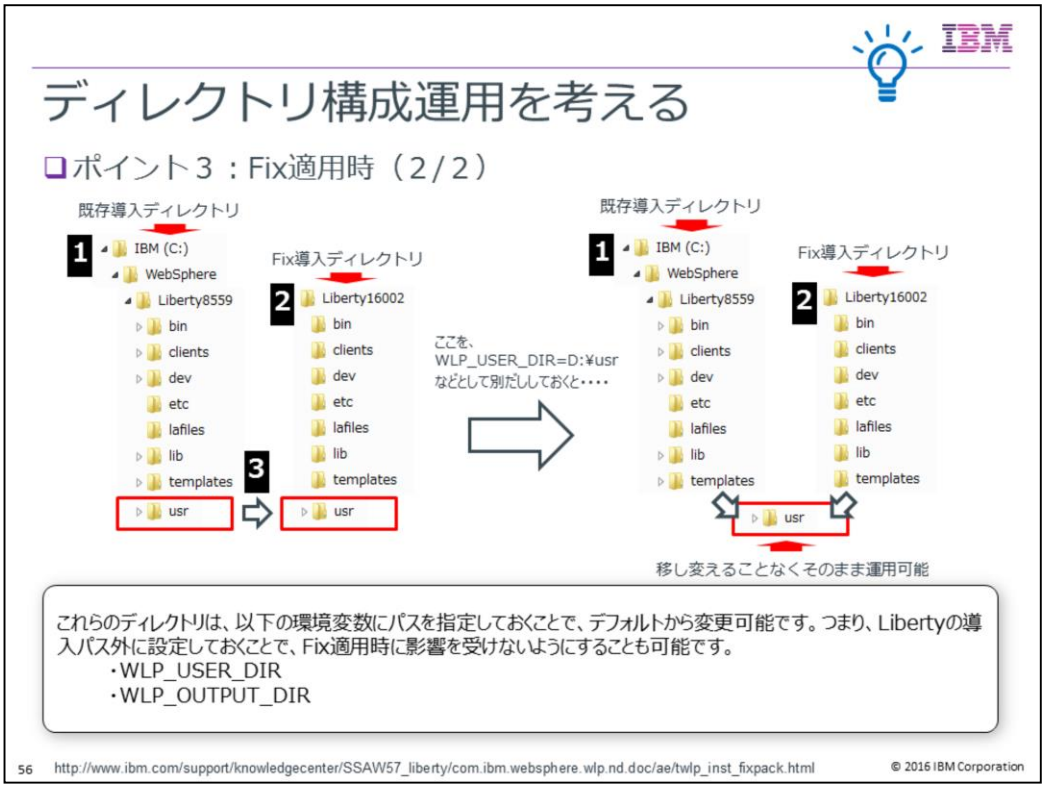


ポイント3つ目は、Fix適用の方法です。これがどのようにディクトリ構成と関わるか、というと、IIMで導入した場合は既存の導入ディレクトリに履歴を保持しながら被せる様に適用可能であるのに対して、アーカイブからの導入は、新規導入と同じようなイメージで別ディレクトリに導入する事になってしまうわけです。（次ページの図参照）

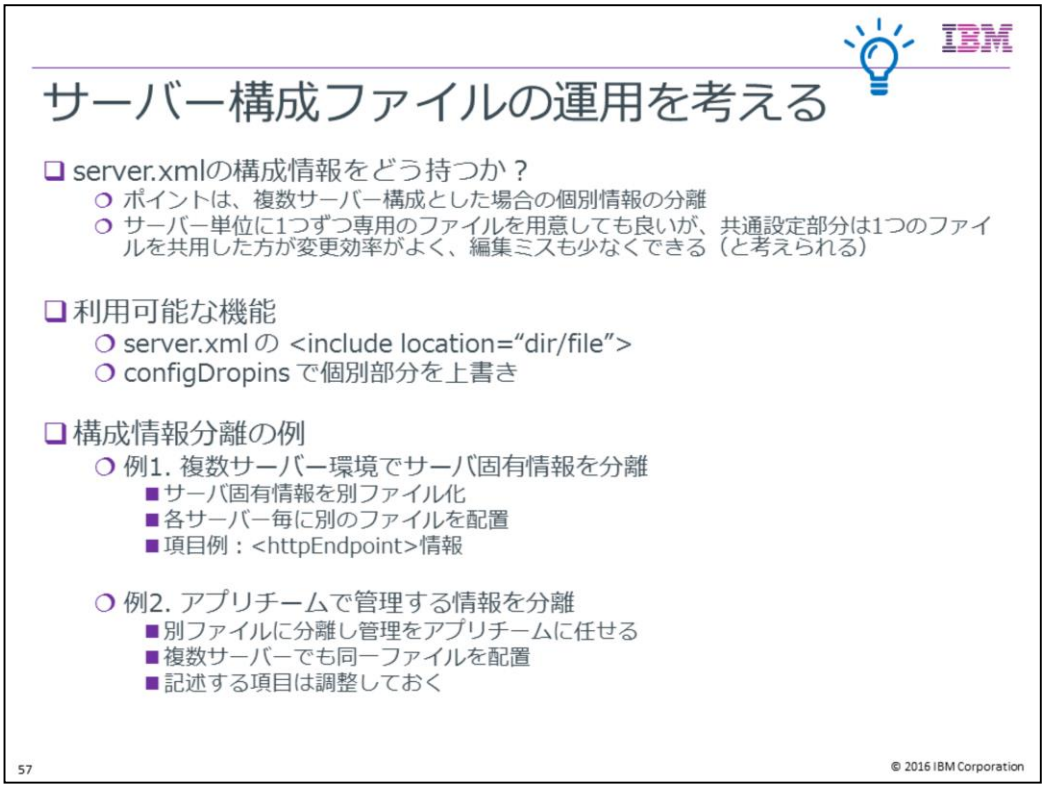
この場合、ポイント2で解説した、ユーザー資産が保管されるディレクトリを、Fix適用（=新規導入）した別ディレクトリに移す、もしくは参照させる方法が必要になります。

そのままファイルをコピーするというのもひとつの方法ではありますが、環境変数経由でこれらのディレクトリを指定することによって、Libertyとは別のディレクトリにログ等のユーザー管理資産を配置する事が可能になります。

こうしておくと、Fix適用時にもファイルコピーなどを行うことなく、新規導入した方のLibertyランタイムの環境変数から参照して、設定なども補完する事が出来ます。



上記は、前頁で説明したFix適用時のディレクトリ指定にイメージ図です。

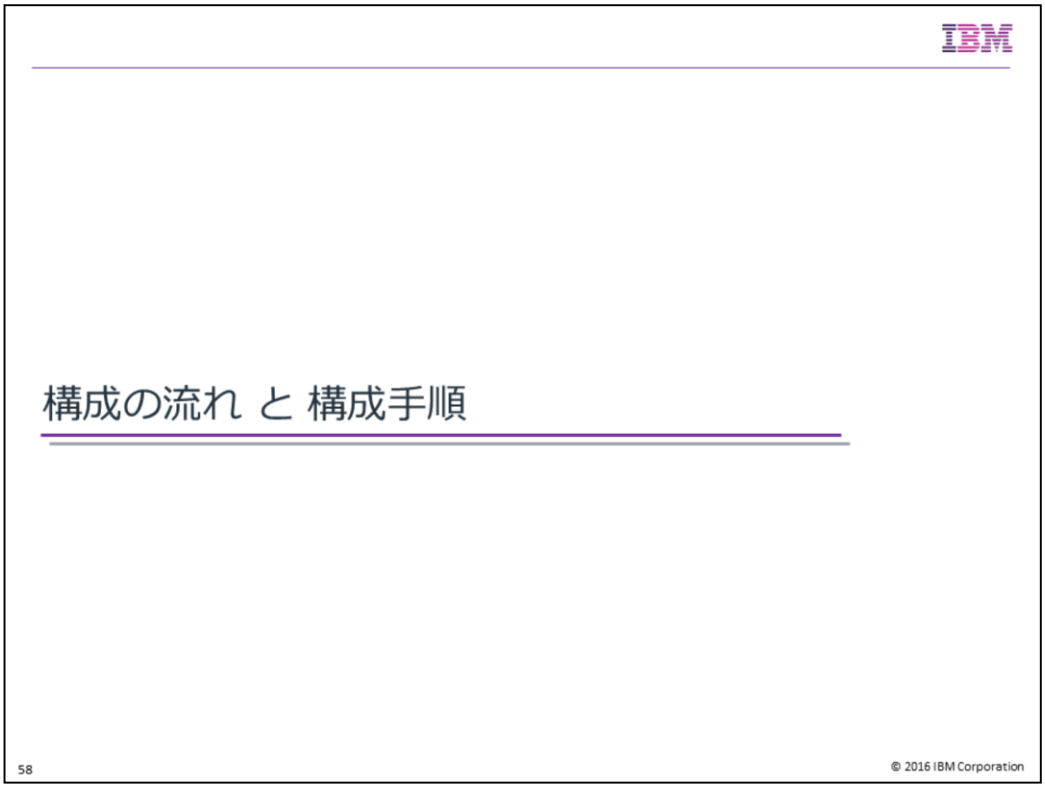


次は、サーバー構成ファイルの運用についてです。

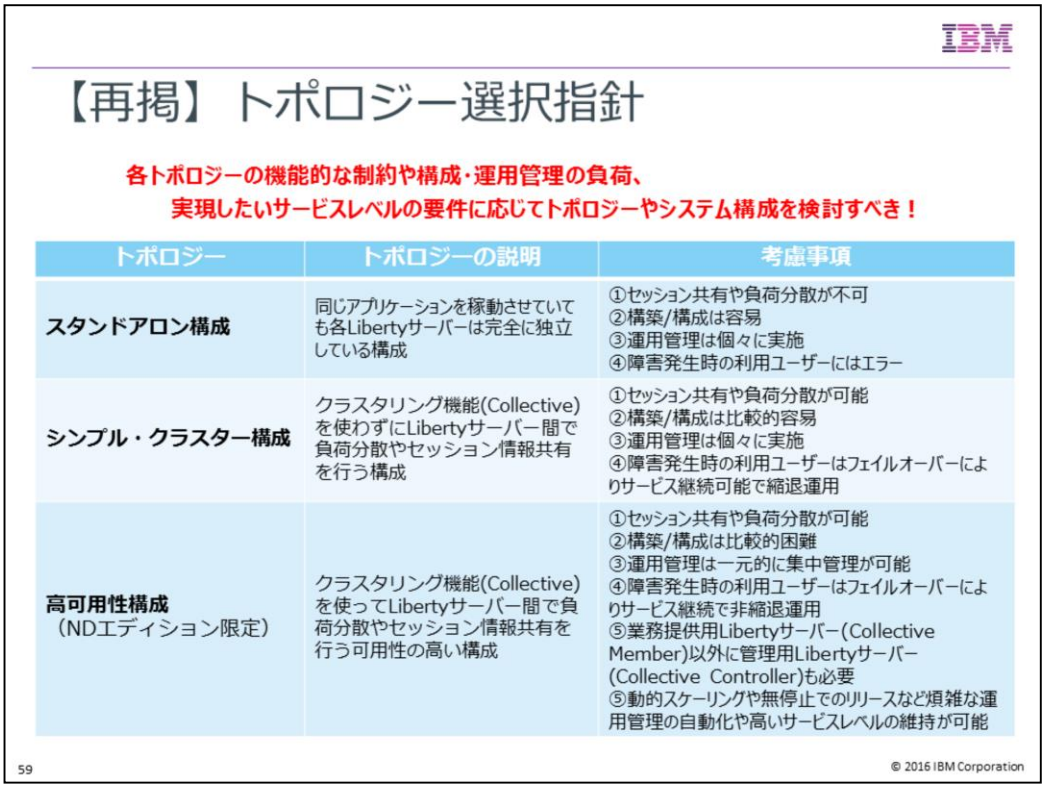
サーバー構成も、include指定することで構成を別ファイルに分離する事が出来たり、configDropinsフォルダへの配置による動的更新など幾つかの便利な機能がありました。

これらを有効活用しつつ、運用のあるべき姿をどう実現していくか、というのを考える必要があります。

例えば、テスト環境を複数持つ場合は環境ごとに異なる部分をincludeで別だしするとか、チーム単位で編集可能な項目を別ファイルに分けて定義して、分散管理を行うとか、様々な方法が考えられます。



ここからは、実際にトポロジーのセッションで紹介したトポロジー例を、どのように構成していくか、流れと手順を説明します。



【再掲】トポロジー選択指針

各トポロジーの機能的な制約や構成・運用管理の負荷、
実現したいサービスレベルの要件に応じてトポロジーやシステム構成を検討すべき！

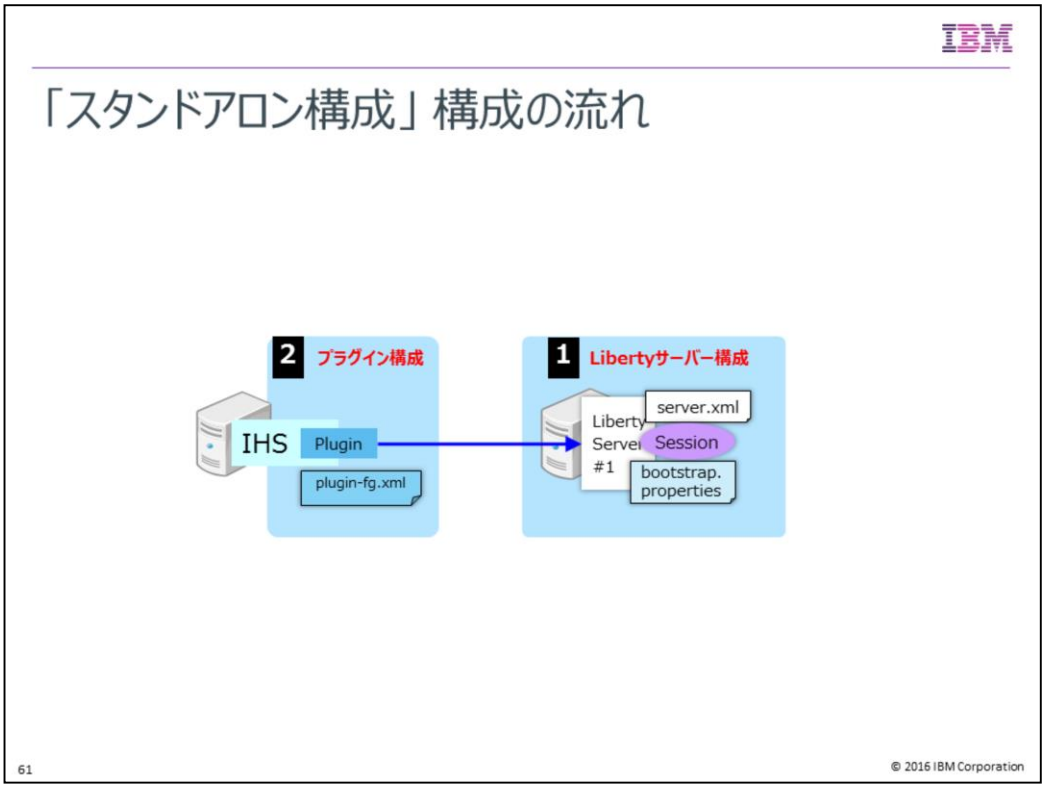
トポロジー	トポロジーの説明	考慮事項
スタンドアロン構成	同じアプリケーションを稼動させていても各Libertyサーバーは完全に独立している構成	①セッション共有や負荷分散が不可 ②構築/構成は容易 ③運用管理は個々に実施 ④障害発生時の利用ユーザーにはエラー
シンプル・クラスター構成	クラスタリング機能(Collective)を使わずにLibertyサーバー間で負荷分散やセッション情報共有を行う構成	①セッション共有や負荷分散が可能 ②構築/構成は比較的容易 ③運用管理は個々に実施 ④障害発生時の利用ユーザーはフェイルオーバーによりサービス継続可能で縮退運用
高可用性構成 （NDエディション限定）	クラスタリング機能(Collective)を使ってLiberty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	①セッション共有や負荷分散が可能 ②構築/構成は比較的困難 ③運用管理は一元的に集中管理が可能 ④障害発生時の利用ユーザーはフェイルオーバーによりサービス継続で非縮退運用 ⑤業務提供用Libertyサーバー(Collective Member)以外に管理用Libertyサーバー(Collective Controller)も必要 ⑤動的スケーリングや無停止でのリリースなど煩雑な運用管理の自動化や高いサービスレベルの維持が可能

59

トポロジーセッションで紹介したトポロジー例を再掲します。

スタンドアロン、シンプル・クラスター、高可用性構成、の3つを挙げて解説します。

1. スタンドアロン構成



Libertyプロファイルでスタンドアロン構成をとる場合の流れを説明します。

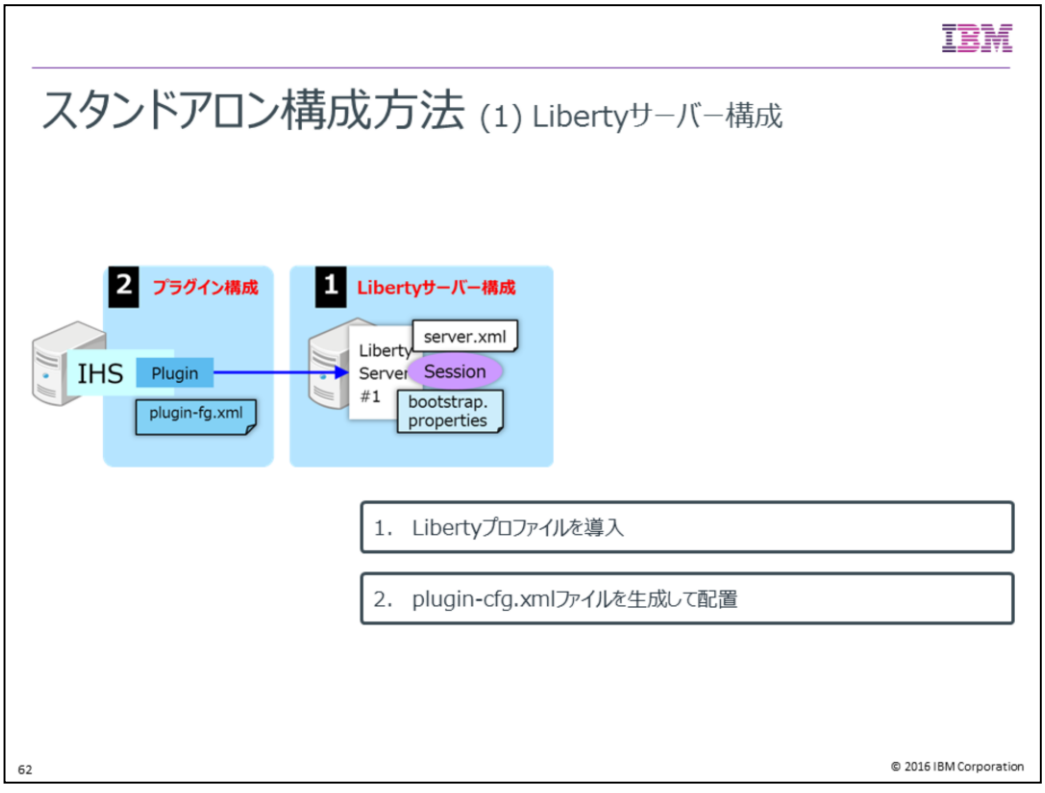
まず、フルプロファイルと同様、通常通りにLoad Balancer/IHS/Pluginといった Libertyサーバー前段のコンポーネントを導入します。

以降の大まかな作業の流れは以下の通りです。

（1）Libertyサーバーを稼動させるノード上にLibertyプロファイルを導入して Libertyサーバーを作成し、サーバーの設定を行います。

（2）プラグインを構成します。

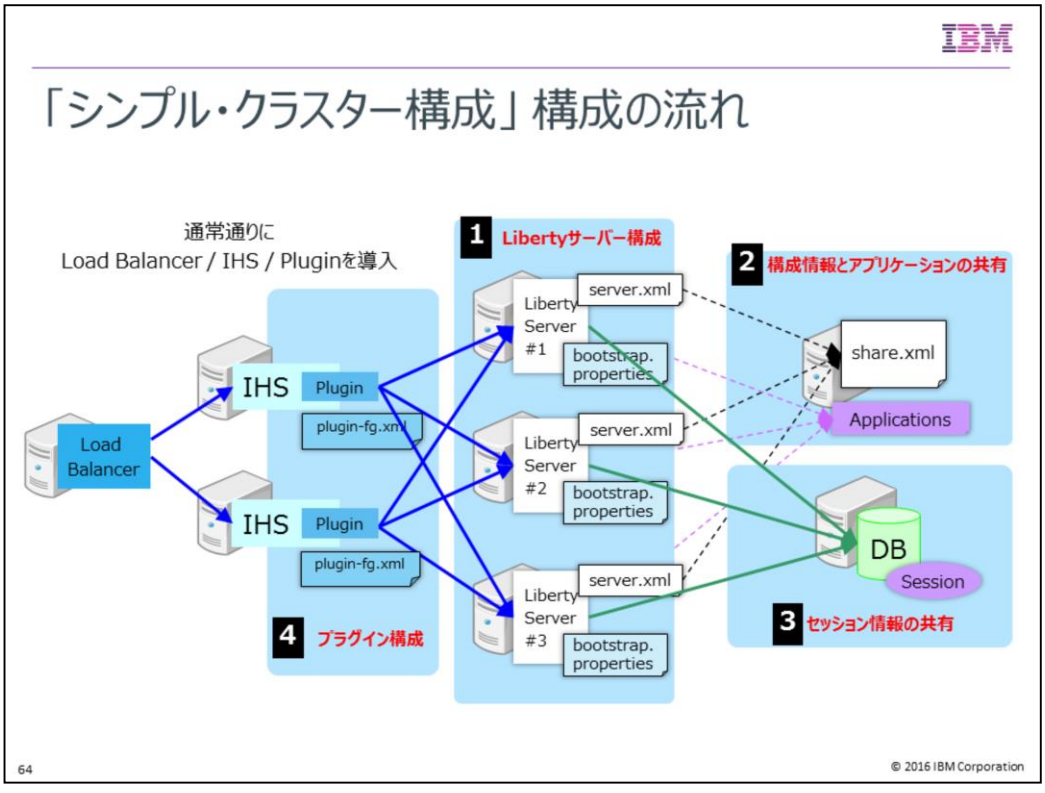
これらの各ステップに関する詳細な手順は次ページ以降でご説明します。



こちらは、Libertyサーバーを稼動させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、サーバー固有の設定をする手順の例です。

1. Libertyサーバー上に Libertyプロファイルを導入します。
2. Libertyサーバーを作成します。
3. server.xmlに対して必要な構成を行います。
4. WDTやmbeanのオペレーションを実行することでplugin-cfg.xmlを生成します。
5. plugin-cfg.xmlをIHSサーバー上の規定の場所に配置します。

2. シンプル・クラスター構成



Libertyプロファイルでシンプル・クラスターを構成する場合の構成方法についてご説明します。

まず、フルプロファイルと同様、通常通りにLoad Balancer/IHS/Pluginといった Libertyサーバー前段のコンポーネントを導入します。

以降の大まかな作業の流れは以下の通りです。

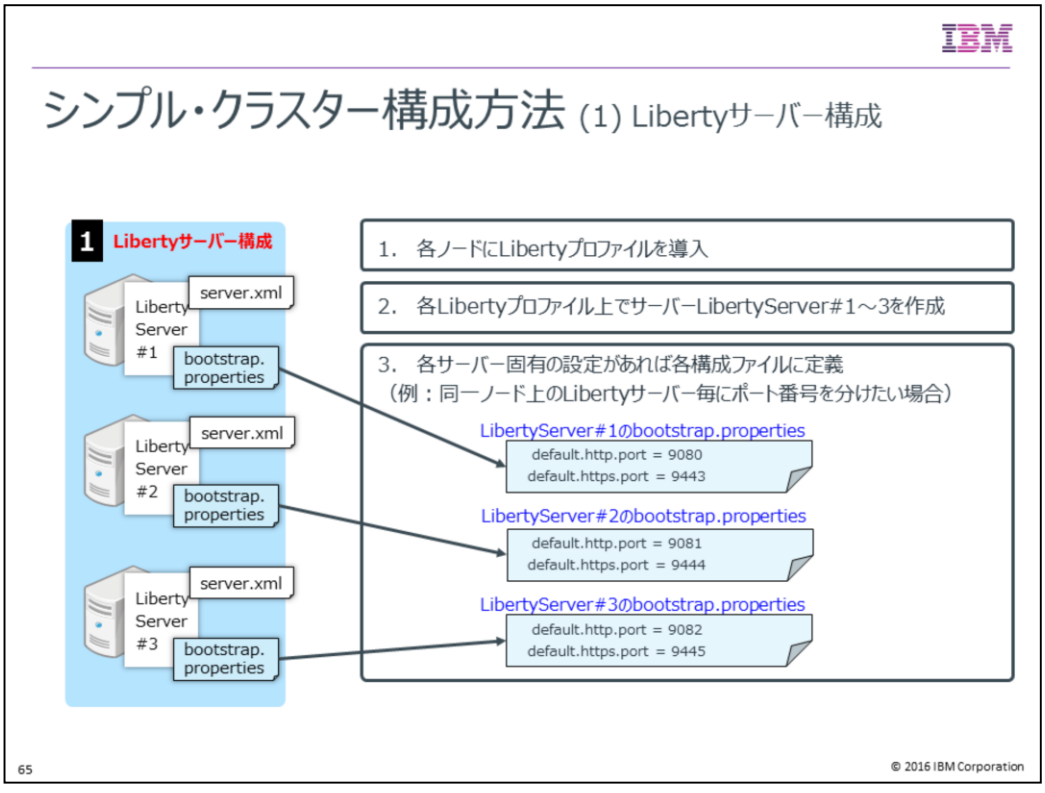
（1）Libertyサーバーを稼動させるノード上にLibertyプロファイルを導入して Libertyサーバーを作成し、サーバー固有の設定をします。

（2）構成変更時の管理対象を最小化するために、Libertyサーバー共通となる構成情報とアプリケーションを共有させるように構成します。

（3）各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成します。

（4）セッション情報を取り扱うアプリケーションの場合は、Libertyサーバー同士でセッション情報を共有するように構成します。

これらの各ステップに関する詳細な手順は次ページ以降でご説明します。



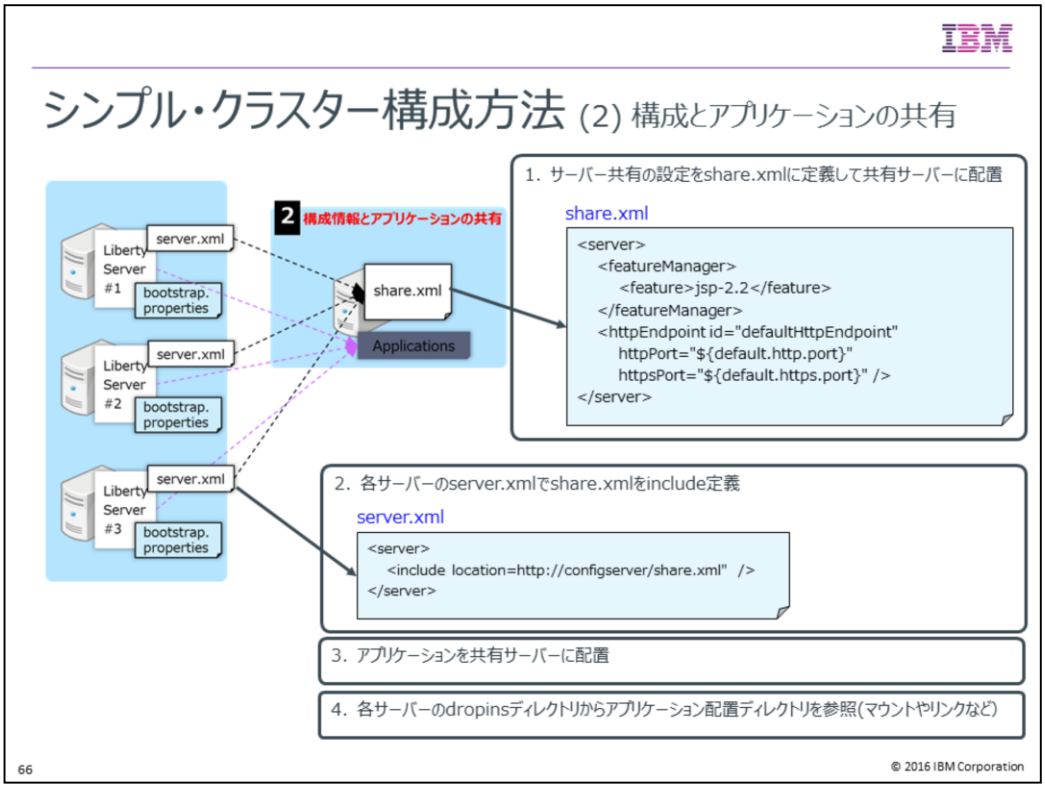
こちらは、Libertyサーバーを稼動させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、サーバー固有の設定をする手順の例です。

1. シンプル・クラスターを構成する全てのノード上にLibertyプロファイルを導入します。

2. 各ノードのLibertyプロファイル上で、LibertyサーバーとしてLibertyServer#1～3を作成します。

3. シンプル・クラスター内のLibertyサーバーで共通的な設定は別の構成ファイルに定義するため、ここでは固有の設定があれば各Libertyサーバーの構成ファイルに定義します。ここでは、例えば同一ノード上に3つのLibertyサーバーを作成している場合、HTTPトランスポートおよびHTTPSトランスポート番号を分ける必要があるため、各Libertyサーバーのbootstrap.propertiesファイルにポート番号を定義している例を挙げています。

bootstrap.propertiesファイルは必須ではないため、作成しない限り存在しません。このファイルはサーバー・ディレクトリーに作成する必要があり、サーバー・ディレクトリーには、構成ルート・ファイル server.xml も含まれています。デフォルトでは、サーバー・ディレクトリーは usr/servers/server_name で、サーバー・ディレクトリーは変更できます。



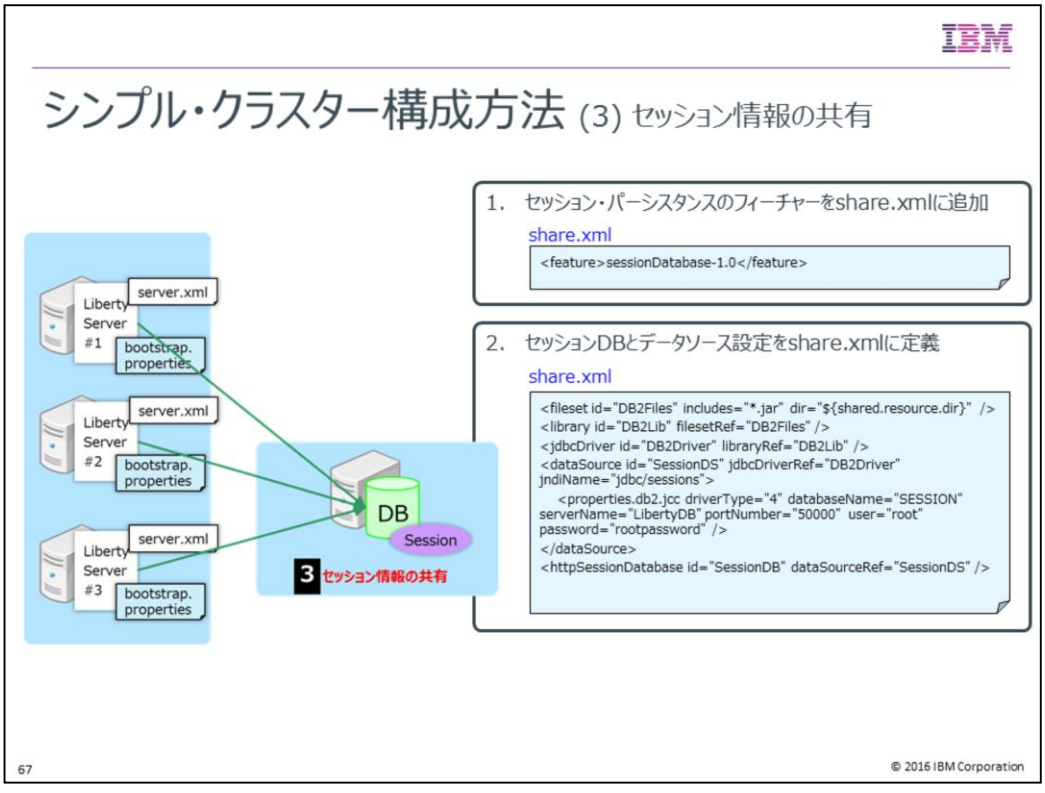
こちらは、構成変更時の管理対象を最小化するために、Libertyサーバー共通となる構成情報とアプリケーションを共有させるように構成する手順の例です。

1. シンプル・クラスター内のLibertyサーバーで共通となる設定をshare.xmlなどserver.xmlとは別の構成ファイルに定義し、ファイルサーバーやWebサーバーなど共有サーバーに配置します。

2. 各Libertyサーバーのserver.xml内にshare.xmlをincludeするように定義します。

3. アプリケーションも共有サーバーに配置します。

4. 各Libertyサーバーのdropinsディレクトリからアプリケーションを配置したディレクトリを参照させるようにOSのマウント等の機能で構成します。

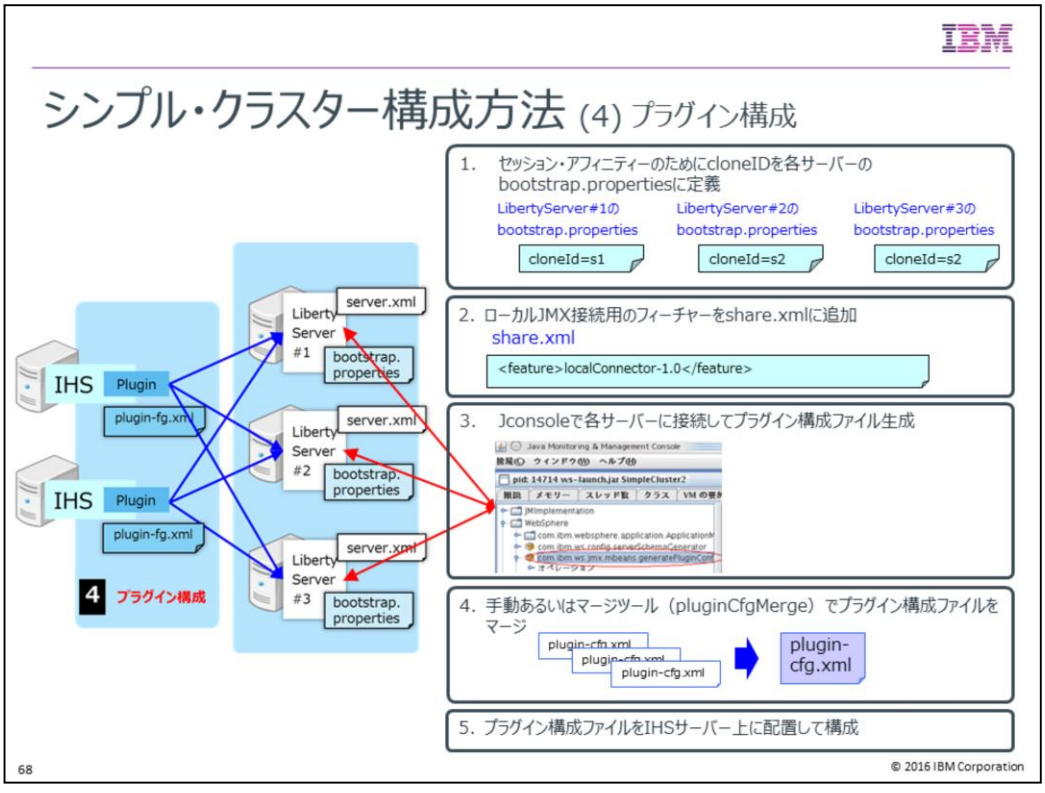


こちらは、セッション情報を取り扱うアプリケーションの場合は、Libertyサーバー同士でセッション情報を共有するように構成する手順の例です。

1. セッション情報をDBに保管するセッション・パーシスタンスを有効にするために、share.xmlファイルにフィーチャー「sessionDatabase-1.0」を追加します。

2. share.xmlファイルに、セッションDBおよびセッションDBに接続するためのデータソースを定義します。

Libertyプロファイルでは、セッション・パーシスタンスのセッション保管先として、DBの他にもWebSphere eXtreme Scale やIBM WebSphere DataPower Appliance XC10 V2 caching appliance が選択できます。



こちらは、各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成する手順の例です。

1. セッション・アフィニティーの機能を持たせるために、各Libertyサーバーのbootstrap.propertiesファイルにcloneIDを定義します。

2. プラグイン構成ファイルを生成するために各LibertyサーバーにJMX接続を行う必要があるため、share.xmlにローカルJMX接続用のフィーチャー「localConnector-1.0」を追加します。

3. Jconsoleで各Libertyサーバーに接続し、プラグイン構成ファイルを生成するオペレーションを実行します。

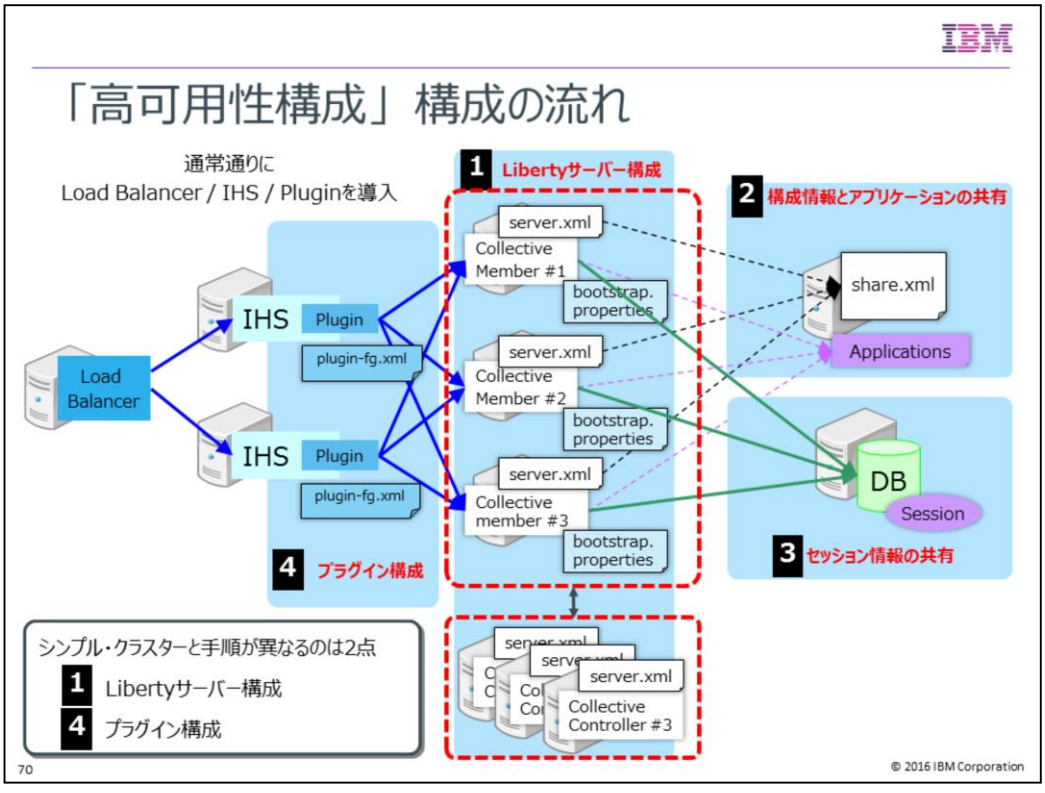
4. 各Libertyサーバーで生成されたプラグイン構成ファイルを手動あるいはフルプロファイルで提供されているマージツールpluginCfgMergeを使用して1つにマージします。

5. マージされたプラグイン構成ファイルをIHSプラグインが読み込めるようにIHSサーバー上に配置します。

＜参考＞

http://www-01.ibm.com/support/docview.wss?uid=swg21674883

3. 高可用性構成



Libertyプロファイルでクラスターを構成する場合の構成方法についてご説明します。

まず、フルプロファイルと同様、通常通りにLoad Balancer/IHS/Pluginといった Libertyサーバー前段のコンポーネントを導入します。

以降の大まかな作業の流れは以下の通りです。

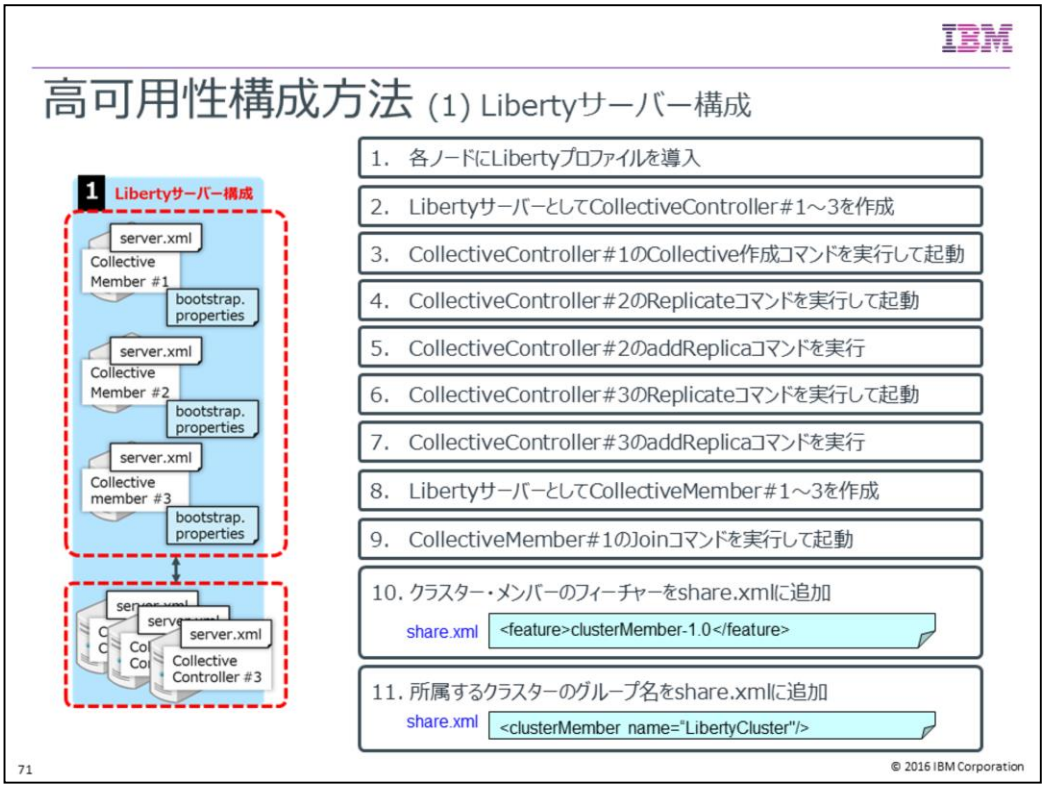
（1）Libertyサーバーを稼動させるノード上にLibertyプロファイルを導入して Libertyサーバーを作成し、Liberty Collective を構成してサーバー固有の設定をします。

（2）構成変更時の管理対象を最小化するために、Libertyサーバー共通となる構成情報とアプリケーションを共有させるように構成します。

（3）セッション情報を取り扱うアプリケーションの場合は、Libertyサーバー同士でセッション情報を共有するように構成します。

（4）各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成します。

これらのステップのうち、シンプル・クラスターの構成手順と異なるのは（1）と（4）のみですので、それらに関する詳細な手順を次ページ以降でご説明します。



こちらは、Libertyサーバーを稼動させるノード上にLibertyプロファイルを導入してLibertyサーバーを作成し、Liberty Collective を構成してサーバー固有の設定をする手順の例です。

1. Liberty Collective を構成する全てのノード上にLibertyプロファイルを導入します。

2. Collective Controller となるLibertyサーバーとしてCollectiveController#1～3を作成します。

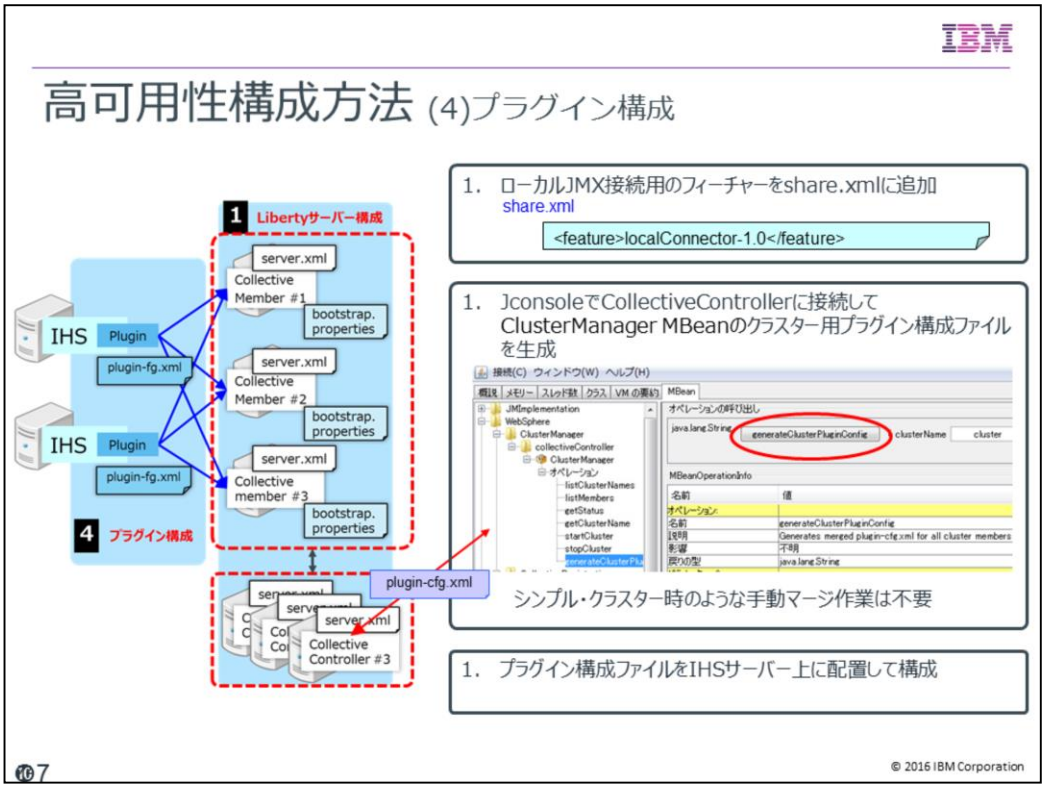
3～7. Libertyプロファイルで提供されているCollective Controller を構成するためのCollective、Replicate、addReplicaコマンドを使用してCollective ControllerとReplica Set を構成します。

8. Collective Member となるLibertyサーバーとしてCollectiveMember#1～3を作成します。

9. Libertyプロファイルで提供されているCollective Member を構成するためのJoinコマンドを使用してCollective Member を構成します。

10. 共通設定を共有するためのshare.xmlファイルに「clusterMember-1.0」フィーチャーを追加します。

11. 更にshare.xmlにCollective Member が所属するクラスターのグループ名を定義します。



こちらは、各Libertyサーバーへの負荷分散やフェールオーバーが行えるように、プラグインを構成する手順の例です。

1. プラグイン構成ファイルを生成するために各LibertyサーバーにJMX接続を行う必要があるため、share.xmlにローカルJMX接続用のフィーチャー「localConnector-1.0」を追加します。

2. Jconsoleでいずれかの Collective Controller に接続し、ClusterManager Mbeanのプラグイン構成ファイルを生成するオペレーションを実行します。ここでは、既にクラスターのメンバー分の定義情報を含んだマージ版のプラグイン構成ファイルが生成されますので、手動でのマージ作業は不要です。

3. 生成されたプラグイン構成ファイルをIHSプラグインが読み込めるようにIHSサーバー上に配置します。

WebSphere Application Server Liberty